COMMODORE 16 per te

Rita Bonelli
Luciano Pazzucconi
Fabio Racchi

BASIC 3.5

GRUPPO EDITORIALE JACKSON

COMMODORE 16 per te

Rita Bonelli
Luciano Pazzucconi
Fabio Racchi

GRUPPO
EDITORIALE
JACKSON
Via Rosellini, 12
20124 Milano

© Copyright per l'edizione originale Gruppo Editoriale Jackson - Febbraio 1985

COORDINAMENTO EDITORIALE: Daria Gianni
COPERTINA: Silvana Corbelli
GRAFICA E IMPAGINAZIONE: Francesca di Fiore
STAMPA: Alberto Matarelli - Milano -
Stabilimento grafico

Tutti i diritti sono riservati. Stampato in Italia. Nessuna parte di questo libro puo' essere riprodotta, memorizzata in sistemi di archivio, o trasmessa in qualsiasi forma o mezzo, elettronico, meccanico, fotocopia, registrazione o altri senza la preventiva autorizzazione scritta dell'editore.

SOMMARIO

CAPITOLO 0: PRELIMINARI
0.1 Di cosa parliamo 1
0.2 Come si collegano calcolatore, video e registratore 2
0.3 Altre possibilita' di collegamento .. 5
0.4 L'unita' DATASSETTE 1531 6
0.5 Come usare la cassetta allegata 8

CAPITOLO 1: LA TASTIERA
1.1 Uno sguardo d'insieme 13
1.2 Tasti alfabetici e numerici 13
1.3 Tasti SHIFT, CBM LOGO e CTRL 14
1.4 Tasti di controllo 15
1.5 Tasti di funzione 19
1.6 Tasto RESET 19
1.7 EDITOR 20
1.8 Riepilogo 23

CAPITOLO 2: IL VIDEO
2.1 Caratteristiche e nomenclatura 25
2.2 I colori e gli effetti speciali 26
2.3 Riepilogo 28

CAPITOLO 3: LA CALCOLATRICE
3.1 Il COMMODORE 16 come calcolatrice ... 29
3.2 Riepilogo 35

CAPITOLO 4: IL PROGRAMMA
4.1 Cosa e' un programma 37
4.2 Il linguaggio BASIC 39
4.3 Come si organizza un programma 41

4.4 Le istruzioni del BASIC 43
4.5 Come si scrive un programma 45
4.6 Le istruzioni piu' elementari
del BASIC 47
4.7 Come si esegue un programma 54
4.8 Riepilogo 55

CAPITOLO 5: GESTIONE DEL PROGRAMMA
5.1 Come si scrive piu' facilmente
un programma 57
5.2 Memorizzazione e caricamento
dei programmi 59
5.3 Riepilogo 63

CAPITOLO 6: STRUTTURE DEL PROGRAMMA
6.1 Struttura condizionale 65
6.2 I cicli 68
6.3 I sottoprogrammi 74
6.4 I salti calcolati 76
6.5 Riepilogo 77

CAPITOLO 7: LA GRAFICA
7.1 Alta risoluzione e multicolore 79
7.2 Il pennino (o cursore grafico) 80
7.3 Sistemi di coordinate 81
7.4 Modi grafici (l'istruzione GRAPHIC) . 81
7.5 I colori nei modi grafici 82
7.6 Le istruzioni grafiche 83
7.7 Punti e rette (l'istruzione DRAW) ... 84
7.8 Circonferenze, ellissi e poligoni
(l'istruzione CIRCLE) 85
7.9 L'istruzione PAINT 86
7.10 Disegnare rettangoli
(l'istruzione BOX) 86
7.11 Scrivere caratteri in modo grafico
(l'istruzione CHAR) 87
7.12 Trasferire parti di schermo
grafico (le istruzioni SSHAPE
e GSHAPE) 87

7.13 Funzioni grafiche 89
7.14 Riepilogo 93

CAPITOLO 8: ANIMAZIONE
8.1 Il movimento 95
8.2 Un esempio di movimento 96
8.3 Riepilogo 97

CAPITOLO 9: IL SUONO
9.1 Il suono 99
9.2 Riepilogo 101

CAPITOLO 10: ALTRE ISTRUZIONI BASIC
10.1 Le variabili con indice 103
10.2 Il trattamento delle stringhe 106
10.3 Lettura dati dall'interno
del programma 111
10.4 L'istruzione GET 114
10.5 Riepilogo 115

CAPITOLO 11: OPERAZIONI AVANZATE
11.1 Le finestre sul video 117
11.2 Definizione delle funzioni utente ... 119
11.3 Alcune funzioni avanzate 120
11.4 Come trovare gli errori
nel programma 122
11.5 La gestione degli errori
nel programma 123
11.6 Gestione file di dati su
cassetta 125
11.7 Esempio archivio di dati
su cassetta 134
11.8 Riepilogo 150

APPENDICE A: IL BASIC 3.5

A.1 Introduzione 151
A.2 Costanti e variabili 154

A.3 Operatori aritmetici, relazionali e logici 159
A.4 Comandi, istruzioni, funzioni 161

APPENDICE B: CODICI E NUMERI DEL CALCOLATORE 217

APPENDICE C: UTILIZZO DELLA MEMORIA 249

APPENDICE D: VALORE DELLE NOTE 265

APPENDICE E: MESSAGGI DI ERRORE 269

APPENDICE F: FUNZIONI MATEMATICHE DERIVATE 279

APPENDICE G: CONVERSIONI DECIMALE, ESADECIMALE, BINARIO 281

RINGRAZIAMENTI

Gli autori ringraziano James Bachmann, Amministratore Delegato, e Sergio Messa, Direttore Generale, della Commodore Italiana s.p.a., per aver messo a disposizione le apparecchiature e la documentazione necessarie alla realizzazione dell'opera.

Nel testo per esigenze tipografiche la FRECCIA VERSO L'ALTO, che ha il significato di elevamento a potenza, e' stampata col carattere ^.

PREFAZIONE

Eccoci pronti a parlarti di un nuovo calcolatore, il COMMODORE 16, al quale auguriamo, con la certezza che l'avra', buona fortuna!

Abbiamo voluto scegliere un titolo molto amichevole e un metodo molto semplice per farti cominciare a prendere confidenza con questo calcolatore, che per te potrebbe essere il primo. Anzi, il nostro principale intento, quando abbiamo progettato questo libro, e' stato quello di aiutare un principiante a imparare ad usare il calcolatore; se tu sei gia' un esperto di informatica ti servono solo quelle parti di esso che descrivono il BASIC 3.5.

Il libro si propone di insegnare ad usare il COMMODORE 16 e a programmarlo in BASIC, servendosi della tastiera, del video e del registratore a cassetta.
La gestione delle altre periferiche, come stampante e unita' disco, gli approfondimenti del BASIC e del SISTEMA OPERATIVO e la programmazione in ASSEMBLER sono argomenti che saranno oggetto del secondo volume, "COMMODORE 16 sempre di piu'", che fara' seguito a questo primo.

La cassetta contiene la presentazione degli argomenti che sono trattati nei capitoli da 1 a 10. Ti suggeriamo di seguire il programma sul video, prima di leggere il relativo capitolo, e di rifletterci sopra.

A volte siamo stati costretti ad anticipare dei concetti per proseguire nel discorso, ma abbiamo cercato di spiegare tutto quello che diciamo. Ti chiediamo di darci fiducia e di seguire il libro dall'inizio, rispettando le nostre indicazioni. Ci auguriamo che alla fine tu sia contento e sia entrato a far parte della schiera di persone che, come noi, apprezzano il calcolatore elettronico.

Ti consigliamo di iniziare con la lettura del Capitolo 0. I capitoli da leggere dopo aver seguito i relativi programmi, contenuti sulla cassetta, sono quelli da 1 a 10. Alla fine di ogni capitolo e' presente un riepilogo molto schematico sui concetti che riteniamo tu debba aver imparato nel capitolo stesso.

Alla fine del Capitolo 11 descriviamo un programma per gestire un archivio di dati su cassetta.
L'Appendice A contiene la scheda completa del BASIC 3.5. Completano la trattazione le Appendici da B a G. L'Appendice E tratta in modo approfondito le segnalazioni di errore da parte del sistema. In alcune appendici il linguaggio e' meno divulgativo e un po' piu' tecnico, pertanto te ne raccomandiamo la lettura solo dopo aver assimilato il contenuto dei capitoli precedenti.

Non ci resta che augurarti: BUON LAVORO!

Gli autori

CAPITOLO 0

PRELIMINARI

0.1 DI COSA PARLIAMO

In questo libro parliamo del COMMODORE 16; che e' un calcolatore della famiglia dei personal, cioe' un calcolatore di modesto costo e dimensioni, ma piuttosto potente.

Le principali parti componenti il nostro calcolatore sono:
. unita' centrale, cioe' il calcolatore,
. unita' di ingresso principale, la tastiera,
. unita' di uscita principale, il video.

L'unita' centrale comprende il microprocessore base del calcolatore, chiamato anche CPU (Central Processing Unit), e tutta l'elettronica necessaria al funzionamento delle periferiche, tastiera e video. Tra l'altro esso comprende le parti necessarie per il collegamento del registratore, periferica esterna per l'ingresso e l'uscita dei dati su cassetta magnetica.
In generale, viene acquistato il registratore insieme al calcolatore, per avere la possibilita' di leggere e memorizzare programmi.

Il calcolatore e le periferiche sono l'HARDWARE, la parte fisica delle apparecchiature. Il calcolatore non funziona se non si dispone anche del SOFTWARE, cioe' dei programmi. I programmi, registrati su un supporto magnetico come le cassette, si possono comprare e si possono preparare da soli, dopo aver imparato.

Il calcolatore elettronico e' una macchina che funziona eseguendo programmi. I divertenti videogiochi sono dei programmi.

Una parte importante dello hardware del calcolatore e' la MEMORIA, cioe' quella parte che permette di conservare programmi e dati e quindi di lavorare.
Nel COMMODORE 16 una parte di memoria e' stata preventivamente trattata e contiene dei programmi registrati in modo indelebile; essa viene chiamata memoria ROM (Read Only Memory). La parte di memoria dove puoi scrivere e leggere viene siglata con RAM.
Il calcolatore viene quasi sempre collegato ad altre apparecchiature; si chiamano INTERFACCE i dispositivi che consentono di collegare tra loro componenti diverse.

0.2 COME SI COLLEGANO CALCOLATORE, VIDEO E REGISTRATORE

Hai davanti a te le unita' che compongono il tuo COMMODORE 16, vediamo cosa sono.
La TASTIERA contiene il CALCOLATORE ed e' il principale mezzo per inserire i dati; hai in un blocco solo il calcolatore e una sua unita' periferica, la tastiera. Si chiama UNITA' PERIFERICA una apparecchiatura che si collega al calcolatore esternamente; nel caso della tastiera il collegamento e' gia' fatto ed e' fisso.
Il REGISTRATORE e' una periferica che serve per immettere dati nel calcolatore e per riceverli; su di essa si inserisce il supporto fisico CASSETTA di NASTRO MAGNETICO. Il registratore della COMMODORE adatto al modello 16 si chiama DATASSETTE 1531.
Il VIDEO e' la principale periferica per l'uscita dei dati; si puo' usare un normale televisore o un MONITOR.

Prima di iniziare i collegamenti ti consigliamo di osservare con attenzione il lato destro del calcolatore e la sua parte posteriore.

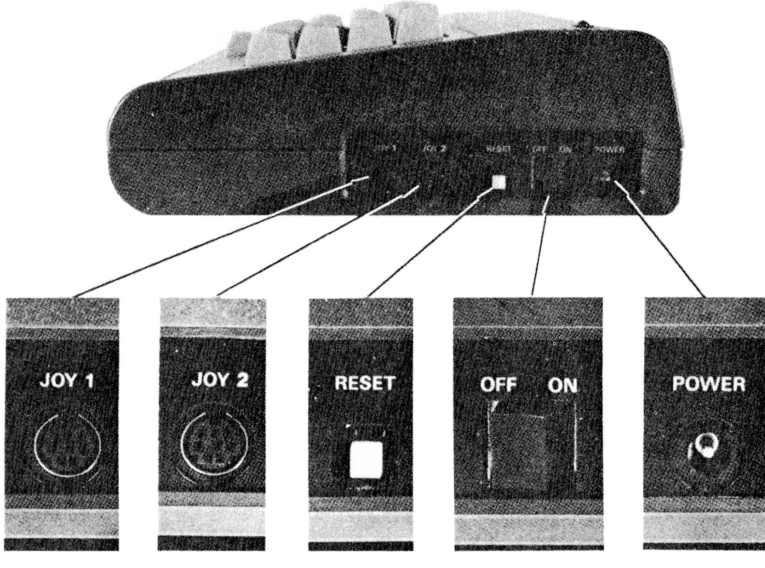


Figura 0.1 Parte laterale destra tastiera

Riferendoci alla Figura 0.1 prendiamo ora in esame solo quanto ci serve per collegare calcolatore, registratore e video. Partendo da destra, la PRESA POWER serve per collegare il cavetto dell'ALIMENTATORE, il quale dovra' essere collegato alla presa di corrente. L'INTERRUTTORE, marcato OFF/ON, serve per togliere o dare corrente al calcolatore. Quando il calcolatore e' alimentato da corrente si accende in rosso la spia posta sulla tastiera in alto a destra vicino alla scritta POWER. Il PULSANTE GRIGIO CHIARO, marcato RESET, serve per ripristinare le condizioni iniziali del calcolatore, come al momento dell'accensione.

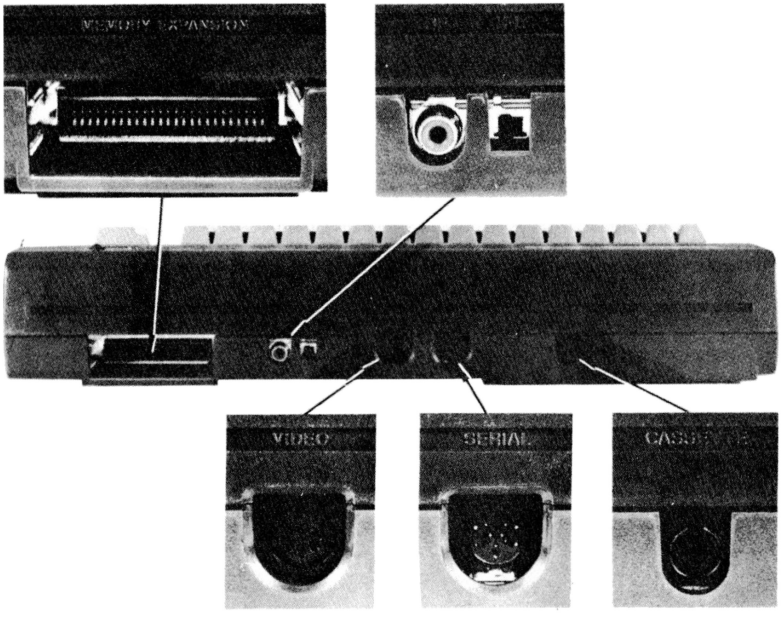


Figura 0.2 Parte posteriore tastiera

Sul lato posteriore vedi quanto riportato in Figura 0.2. Partendo da destra la PRESA marcata CASSETTE, serve per inserire il cavo del registratore; vedrai che sullo spinotto e' sovraincisa una freccia a indicare la parte superiore. La PRESA VIDEO serve per collegare il cavo del MONITOR (che non e' contenuto nella scatola), se ti servi di questa apparecchiatura; se usi un normale televisore non devi usare la presa MONITOR. La PRESA RF serve per collegare, con il cavetto contenuto nella scatola, il calcolatore al televisore, se ti servi di questa apparecchiatura.

Per operare correttamente devi assicurarti che l'interruttore del calcolatore sia in posizione OFF, poi devi operare i collegamenti:

. calcolatore-televisore
  o calcolatore-MONITOR,
. calcolatore-registratore,
. calcolatore-alimentatore,
. alimentatore-RETE 220V,

alla fine devi accendere il calcolatore e il televisore e sintonizzare il televisore sul CANALE 36 UHF, come se fosse una TV privata, oppure accendere il MONITOR, a seconda dell'apparecchiatura usata.
A questo punto vedrai apparire sul video la scritta che segue:

COMMODORE BASIC V3.5 12277 BYTES FREE
READY.

e sotto la lettera R di READY un quadratino lampeggiante, chiamato CURSORE.
Il significato di questa scritta ti verra' chiarito in seguito.

0.3 ALTRE POSSIBILITA' DI COLLEGAMENTO

In seguito potrai desiderare di arricchire il tuo calcolatore con le altre periferiche disponibili. Vediamo quali ulteriori collegamenti sono possibili.
Tornando alla Figura 0.1, vediamo a sinistra le due PRESE marcate JOY 1 e JOY 2; esse servono per collegare al COMMODORE 16 due JOYSTICK, che sono dispositivi per l'ingresso di dati. I joystick danno la possibilita' di muovere oggetti disegnati sul video e di emettere un segnale particolare, di norma chiamato fuoco; essi sono quindi molto utili per la programmazione di giochi.
Consideriamo ora la Figura 0.2; in essa abbiamo

trascurato la PRESA marcata SERIAL e la FENDITURA marcata MEMORY EXPANSION. La presa SERIAL puo' essere usata per collegare al COMMODORE 16 una unita' a floppy disk o una stampante. Qualora si vogliano collegare ambedue le unita' periferiche citate, si opera un collegamento in serie, cioe' si collega al calcolatore una delle due unita' e la seconda unita' alla prima.
La periferica che usa come supporto fisico di registrazione i dichetti magnetici, chiamati FLOPPY DISK, si rivela molto utile quando aumenta la quantita' di dati che si vuole gestire con il calcolatore e, inoltre, si desidera poter accedere ai dati stessi con una velocita' superiore a quella consentita dall'uso delle cassette.
La STAMPANTE si rende necessaria quando si vuole poter conservare delle elaborazioni sotto forma di documenti direttamente consultabili.
La MEMORY EXPANSION e' una porta che consente di inserire nel COMMODORE 16 appositi cartridge (blocchi di memoria), che possono contenere programmi registrati o espansioni di memoria.

Devi ricordare che quando il COMMODORE 16 e' alimentato da corrente, cioe' la spia POWER e' rossa, non devi operare collegamenti, altrimenti rischi di danneggiare le tue apparecchiature.

0.4 L'UNITA' DATASSETTE 1531

Questa periferica non deve essere collegata alla tensione di rete, infatti riceve corrente tramite il collegamento al calcolatore.

Osservando il registratore vedi nella parte frontale 6 tasti, al di sopra di essi l'alloggiamento per la cassetta e di fianco il conta-

giri con il pulsante per il suo azzeramento.
Il tasto marcato EJECT serve per aprire l'alloggiamento della cassetta. Il tasto marcato STOP serve per rimettere in posizione di riposo (e quindi disattivare) qualunque tasto abbassato tra i primi 4 a sinistra. Il tasto marcato F.FWD serve per fare avanzare velocemente il nastro; puo' quindi essere usato per posizionarsi manualmente a un certo punto del nastro. Il tasto marcato REWIND serve per riavvolgere il nastro. Il tasto marcato PLAY, se usato da solo fa avanzare il nastro a velocita' di lettura; esso deve essere premuto per poter leggere un nastro. Il tasto marcato RECORD, che quando premuto fa abbassare contemporaneamente anche il tasto PLAY, serve per far avanzare il nastro a velocita' di scrittura e scrivere sul nastro.

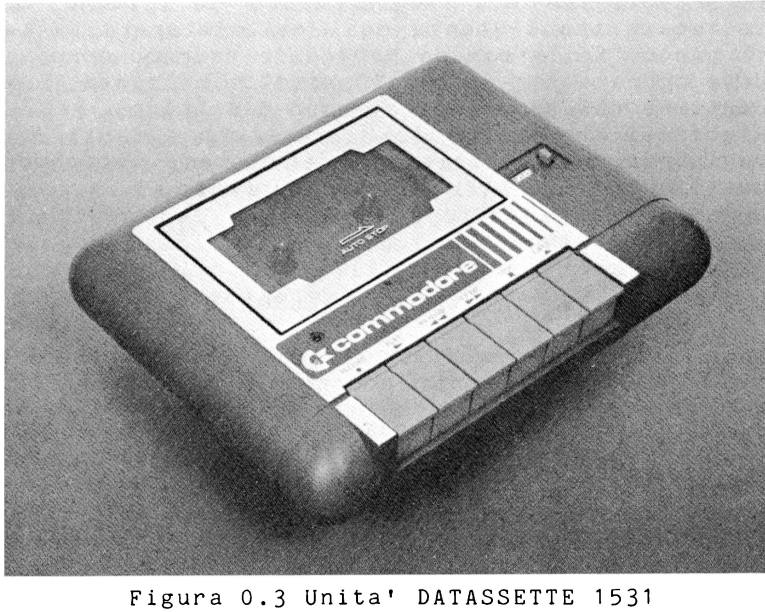


Figura 0.3 Unita' DATASSETTE 1531

L'alloggiamento della cassetta deve essere chiuso manualmente con una leggera pressione dopo l'introduzione della cassetta. Il contagiri puo' essere molto utile se usato correttamente. Ti consigliamo di azzerare il contagiri quando inserisci una cassetta; in tale modo puoi leggere il valore raggiunto dopo ogni memorizzazione e annotarlo, servendoti in seguito di questa informazione per posizionarti dove vuoi sul nastro.
Ti consigliamo di usare cassette corte, al massimo di 30 o 46 minuti. E' naturalmente necessario aver cura delle cassette, proteggerle dalla polvere e tenerle lontane da fonti magnetiche. Anche il registratore va tenuto con cura; esistono in commercio prodotti adatti alla pulizia e alla smagnetizzazione delle testine di lettura e scrittura.

Durante l'uso del registratore si svolge un colloquio tra l'utente e il calcolatore tramite il video. Nel prossimo paragrafo vedremo come si deve operare per usare la cassetta allegata, che contiene dei programmi a scopo didattico.
Ricordati che e' meglio operare con i tasti del registratore in posizione di riposo, azionando quelli richiesti al momento opportuno.

0.5 COME USARE LA CASSETTA ALLEGATA

Dopo aver operato i collegamenti ed acceso il tuo COMMODORE 16 puoi cominciare a conoscerlo. Noi ti consigliamo di inserire nel registratore la cassetta, allegata a questo libro, dal lato A, di riavvolgerla, se necessario, e di scrivere:
LOAD "CAP1" e premere il tasto RETURN
la scritta appare sul video, e, dopo la pressione del tasto RETURN, vedrai apparire sul video la seguente frase:

PRESS PLAY ON TAPE
ti viene cioe' chiesto di premere il tasto marcato PLAY del registratore. Appena hai eseguito compare:
OK
SEARCHING FOR CAP1

dopo poco il video si sbianca e il nastro gira nel registratore. Dopo un po' vedi comparire sul video la scritta:
FOUND CAP1
LOADING

questo significa che sul nastro e' stato ricercato e trovato il programma con nome "CAP1". Dopo un po' il video si sbianca nuovamente e il nastro riprende a girare; questo significa che il programma viene caricato da nastro nella memoria del calcolatore. Quando il caricamento e' terminato, il nastro si ferma e sul video ricompaiono tutte le scritte precedenti con in piu' la parola READY. A questo punto ti consigliamo di NON RILASCIARE il tasto PLAY del registratore, per mezzo del tasto STOP (di norma ti consigliamo di disattivare i tasti del registratore quando e' terminata un'operazione, ma la cassetta dei programmi in certi casi contiene dei programmi registrati a pezzi, che vengono caricati in tempi successivi; se lasci abbassato il tasto PLAY eviti di veder comparire sul video il relativo messaggio di richiesta).
Scrivi poi: RUN e premi il tasto RETURN.
Puoi ora iniziare a seguire sul video il programma CAP1, relativo al Capitolo 1.
Seguire i programmi dimostrativi e' semplice, basta leggere attentamente quello che compare sul video, riflettere e seguire le indicazioni che vengono fornite.
Qualora tu desideri rivedere un programma che e' appena terminato basta scrivere RUN, per tutti i programmi salvo CAP1, CAP2, CAP7 e CAP10, che devono essere ricaricati dopo aver riavvolto il

nastro, e premere il tasto RETURN, dato che il programma e' ancora in memoria.
Se desideri, invece, rivedere un programma precedente a quello seguito per ultimo (o dopo CAP1, CAP2, CAP7 e CAP10), devi riavvolgere il nastro e scrivere:
LOAD "nome"
dove "nome" e' il nome del capitolo che vuoi rivedere, e premere RETURN; quando appare la scritta READY, al termine del caricamento puoi dare il comando RUN per far girare nuovamente il programma. Far "girare" un programma significa eseguirlo. I programmi dimostrativi, formati da piu' moduli, possono essere eseguiti solo partendo dal primo modulo. Inoltre, i programmi composti da piu' moduli sono predisposti per essere letti in sequenza dalla cassetta; se vuoi trasferirli sul dischetto, devi modificare il comando LOAD (lo trovi verso il fondo del programma) in DLOAD.

I nomi dei programmi dimostrativi sono:

CAP1 CAP2 CAP3 CAP4 CAP5
CAP6 CAP7 CAP8 CAP9 CAP10

essi sono registrati in ordine sulla cassetta; per quelli formati da piu' moduli, questi sono registrati in ordine.

E' possibile caricare in memoria un programma da nastro scrivendo solo:
LOAD e premendo il tasto RETURN.
In questo caso si svolge ancora il colloquio sopra descritto, solo che dopo la parola SEARCHING non compare FOR e il nome del programma, mentre dopo la parola FOUND comparira' il nome del primo programma trovato sul nastro, che e' quello che viene caricato.
Nel caso invece si scriva:
LOAD "nome"
ma il nastro sia posizionato molto prima del

programma "nome", nel colloquio tramite video il calcolatore ti segnala su diverse righe FOUND seguito dal nome del programma incontrato, fornendoti cosi' l'elenco di tutti i programmi registrati sul nastro prima di quello che tu stai cercando.
Vedremo piu' avanti come si deve operare per scrivere programmi sulla cassetta, e anche per registrare e leggere dati.

CAPITOLO 1

LA TASTIERA

1.1 UNO SGUARDO D'INSIEME

La prima volta che hai visto la tastiera di un calcolatore, hai notato probabilmente l'analogia con quella di una macchina da scrivere, e in effetti quella del tuo COMMODORE 16 assomiglia molto alla tastiera di una macchina da scrivere. Una tastiera cosi' e' davvero comoda ed efficiente; pochi sono i calcolatori di questo prezzo, e di queste prestazioni, che hanno una tastiera cosi' "professionale".
Il tuo COMMODORE 16 ha una tastiera composta di 66 tasti: 62 "normali" e 4 "di funzione". I 4 tasti "di funzione" sono in realta' tasti a cui tu puoi assegnare il significato che vuoi (ma questo lo vedremo bene nel seguito); i 62 "normali" sono tasti che hanno ciascuno un significato, che tu non puoi cambiare, e che sono usati principalmente per introdurre nel calcolatore i dati relativi ai programmi, ai numeri, ai comandi, ecc.
Poiche' vi sono nella tastiera 3 tasti che svolgono la stessa funzione, 2 tasti SHIFT e un tasto SHIFT LOCK, in realta' vi sono 60 funzioni diverse che sono svolte da 62 tasti.

1.2 TASTI ALFABETICI E NUMERICI

I tasti della tastiera possono essere divisi in diversi tipi, a seconda della funzione che svol-

gono; i tasti con i numeri, ad esempio, li possiamo chiamare "numerici", e quelli con le lettere dell'alfabeto, "alfabetici". Possiamo chiamare "alfanumerici" l'insieme dei tasti numerici e alfabetici, piu' i segni di punteggiatura, spazio, parentesi, #, dollaro, apici, ^, ecc.

1.3 TASTI SHIFT, CBM LOGO e CTRL

Vi sono dei tasti che da soli non svolgono nessuna funzione, e che vanno premuti insieme ad altri, per cambiarne il significato; essi sono SHIFT, CBM LOGO (il tasto in basso a sinistra, sotto RUN/STOP), e CTRL. Possiamo considerare i precedenti come "tasti di servizio".

Il COMMODORE 16 dispone di 2 SET di caratteri in alternativa tra loro, cioe' ne puo' essere attivo uno solo per volta. Al momento dell'accensione e' attivo il SET MAIUSCOLO/GRAFICO, nel quale le lettere compaiono in maiuscolo e si possono ottenere i caratteri grafici con l'uso di SHIFT (quelli posti a destra sul tasto) e di CBM LOGO (quelli posti a sinistra sul tasto). L'altro SET si chiama MINUSCOLO/MAIUSCOLO; quando esso e' attivo le lettere compaiono in minuscolo. Per ottenere le maiuscole si deve usare lo SHIFT, i caratteri grafici che ottenevi con SHIFT nell'altro set non sono piu' disponibili, mentre si ottengono con il tasto CBM LOGO quelli posti a sinistra sul tasto.
Premendo insieme CBM LOGO e SHIFT si passa dal set maiuscolo/grafico a quello minuscolo/maiuscolo, e viceversa.

Il tasto CBM LOGO usato insieme ai tasti numerici da 1 a 8, fa cambiare il colore del cursore nel colore stampato in basso sul tasto premuto.
Usato con gli altri tasti, CBM LOGO produce gli

stessi effetti che lo SHIFT.
Il tasto CBM LOGO ha un'altra importante funzione: usato mentre il calcolatore sta stampando sul video, rallenta notevolmente l'operazione di SCROLLING, in modo da permettere la lettura delle scritte che scorrono.
Il tasto CTRL (control) va premuto, come SHIFT e CBM LOGO, insieme al tasto con cui si vuole usare. Principalmente con CTRL vanno premuti i tasti numerici da 1 a 8, per cambiare il colore del cursore nel colore stampato in alto sul tasto premuto, o i tasti 9 e 0, per inserire e disinserire l'effetto REVERSE. CTRL puo' essere usato anche in unione ai tasti , (virgola) e . (punto) per inserire e disinserire l'effetto FLASH.

Ecco la corrispondenza tasti/colori in italiano:

Tasto	con CBM LOGO			con CTRL		
1	ORNG	=	ARANCIONE	BLK	=	NERO
2	BRN	=	MARRONE	WHT	=	BIANCO
3	YL GRN	=	GIALLO-VERDE	RED	=	ROSSO
4	PINK	=	ROSA	CYN	=	CIANO
5	BL GRN	=	VERDE-BLU	PUR	=	PORPORA
6	L BLU	=	BLU CHIARO	GRN	=	VERDE
7	D BLU	=	BLU SCURO	BLU	=	BLU
8	L GRN	=	VERDE CHIARO	YEL	=	GIALLO

1.4 TASTI DI CONTROLLO

Alcuni tasti non stampano un carattere, ma svolgono una particolare funzione di controllo del cursore; li possiamo chiamare tasti di "controllo". Appartengono a questa categoria i tasti RETURN, CLR/HOME, INST/DEL, RUN/STOP, e i tasti che spostano il cursore lungo lo schermo (i 4 con le frecce), e ESC.

TASTO RETURN
Il tasto RETURN serve per terminare la linea corrente, e memorizzarla. Se la linea non inizia con un numero, il suo contenuto viene subito interpretato come comando BASIC e mandato in esecuzione. Se invece inizia con un numero, essa viene memorizzata nell'area di memoria riservata al programma. Questo tasto puo essere "pericoloso", in quanto, premuto inavvertitamente, puo' far entrare nel programma linee indesiderate, o cancellare linee che gia' esistono; e' quindi buona norma non premere RETURN in continuazione, sopra uno schermo pieno di scritte, ma conviene piuttosto usare i tasti di movimento cursore e scendere dove lo schermo e' pulito. Se non vuoi che la linea corrente venga interpretata, puoi premere SHIFT-RETURN; questa operazione ti permette di terminare la linea corrente senza ne' memorizzarla, ne' mandarla in esecuzione: essa resta sul video.

TASTO CLEAR/HOME
Il tasto CLEAR/HOME ha due significati; usato da solo porta il cursore alla posizione HOME, cioe' nell'angolo in alto a sinistra, usato insieme allo SHIFT cancella tutto lo schermo, e porta il cursore nell'angolo in alto a sinistra.

TASTO INST/DEL
Il tasto INST/DEL, come il tasto CLEAR/HOME, ha due significati. Usato da solo svolge la funzione DELETE, cioe' cancella il carattere a sinistra del cursore, spostando a sinistra tutti gli eventuali caratteri presenti, sulla stessa linea, a destra del cursore. Usato insieme allo SHIFT invece significa INSERT, e inserisce uno spazio tra il carattere che precede il cursore e quello sotto il cursore. Ovviamente i caratteri che si trovano à destra del cursore sono automaticamente spostati a destra, per fare posto allo spazio appena inserito. Per tutti gli

spazi inseriti viene automaticamente attivato il modo "INSERT", che ti permette di inserire caratteri di controllo senza mandarli subito in esecuzione (vedi Paragrafo 1.7)

TASTI DI MOVIMENTO CURSORE
Sono i tasti che si trovano in alto sulla tastiera, a destra dei tasti numerici. La loro funzione e' quella di spostare il cursore lungo lo schermo, senza cancellare quello che vi si trova sotto. L'EDITOR del COMMODORE 16 (programma del sistema che facilita il colloquio video/tastiera) e' molto potente, e ti permette di correggere le linee semplicemente passandovi sopra con il cursore, effettuando le correzioni, e premendo RETURN alla fine.

TASTO RUN/STOP
Questo tasto ha tre diverse funzioni:
. Premuto da solo, durante l'esecuzione di un programma, lo arresta, e fa si' che venga emesso il messaggio BREAK IN ... (numero linea dell'arresto). Il programma puo' riprendere con l'istruzione CONT (continua). Se dopo l'arresto del programma e' stata introdotta qualche nuova linea, o eseguita un'istruzione CLR, allora la ripresa del programma e' impossibile, e viene emesso il messaggio CAN'T CONTINUE.
. Premuto insieme a SHIFT fa si' che venga caricato il primo programma da disco, e vada in esecuzione automaticamente.
. Premuto insieme a RESET arresta il programma e fa entrare in MONITOR (vedi Paragrafo 1.6).

TASTO ESC (ESCAPE)
Per ultimo descriviamo il tasto ESC (escape). Svolge delle funzioni molto potenti, e puo' tornare utile imparare almeno quelle piu' importanti, per risparmiare tempo in fase di scrittura programmi.
A differenza di SHIFT, CBM LOGO, e CTRL, questo tasto non deve essere premuto insieme al tasto

con cui va usato, ma va premuto prima. Ecco la lista dei tasti che possono essere premuti dopo ESC, e l'effetto che producono:

A: Inserimento automatico; quando si scrive su una riga, il carattere che si trova sotto il cursore viene spostato a destra, anziche' essere cancellato.
C: Cancella inserimento automatico.
D: Cancella tutta la linea dove si trova il cursore.
I: Inserisce una linea.
J: Porta il cursore all'inizio della linea corrente.
K: Porta il cursore alla fine della linea corrente.
L: Abilita lo "scrolling", movimento automatico dello sfondo verso l'alto, per liberare l'ultima linea in basso.
M: Disabilita lo "scrolling".
N: Riporta lo schermo a 25 righe, 40 colonne.
O: Cancella il modo QUOTE e INSERT. Nel modo QUOTE si entra dopo aver aperto le virgolette; come si entra nel modo INSERT e' gia' stato spiegato. In questi modi i caratteri di controllo cursore e colore sono stampati come caratteri in campo inverso, anziche' produrre subito il loro effetto. ESC O cancella l'effetto FLASH e REVERSE (vedi Paragrafo 1.7).
P: Cancella la linea corrente dall'inizio al cursore.
Q: Cancella la linea corrente dal cursore alla fine.
R: Riduce lo schermo a 23 righe, 38 colonne. Puo' servire per quei televisori che non riescono a visualizzare tutta la grandezza dello schermo.
T: Posiziona l'angolo sinistro in alto della finestra video (vedi Paragrafo 11.1).
B: Posiziona l'angolo in basso a destra per la finestra video (vedi paragrafo 11.1).
V: Fa scorrere le scritte dello schermo (scrolling) di una riga verso l'alto.

W: Fa scorrere le scritte dello schermo di una riga verso il basso.
X: Non produce alcun effetto; premi X se hai premuto ESC inavvertitamente, e non vuoi eseguire nessuna delle funzioni di ESCAPE.
E' possibile ottenere le funzioni di escape anche da programma, mediante l'istruzione PRINT CHR$(27). Ad esempio, per ottenere l'inserimento automatico:
PRINT CHR$(27)"A".

1.5 TASTI DI FUNZIONE

Hai notato sicuramente i 4 tasti a destra, con le scritte f1, f2, ... f7, HELP.
Questi tasti sono molto comodi, poiche' ad essi puo' essere assegnato il significato che vuoi.
Per sapere quale significato hanno ora questi tasti, puoi digitare l'istruzione:
KEY
e premere RETURN.

Il calcolatore scrivera' sul video i significati attuali dei tasti funzione.
Se invece vuoi assegnare al tasto f1 una determinata funzione, puoi digitare:
KEY 1,"COMMODORE 16"
e poi premere RETURN.
In questo modo, quando premi f1 appare sul video la scritta COMMODORE 16. Ovviamente, per far tornare i tasti alle loro funzioni originali, puoi assegnare nuovamente a ciascun tasto la funzione precedente, o premere il tasto di RESET, che si trova vicino all'interruttore di alimentazione.

1.6 TASTO RESET

Il tasto RESET e' l'unico che non si trova sulla tastiera; esso infatti e' situato vicino

all'interruttore di alimentazione, sul lato destro del tuo COMMODORE 16. Premere questo tasto assomiglia un po' a spegnere e riaccendere il calcolatore, ma con alcune differenze:
. Il contenuto della memoria utente non viene alterato, anche se i puntatori del BASIC vengono riportati nella condizione iniziale.
. Tenendo premuto RUN/STOP e premendo RESET si arresta il programma corrente, anche se una imprudente routine di TRAP (vedi Paragrafo 11.4) lo impedisce normalmente. Questa operazione ti porta in MONITOR; per tornare al BASIC basta premere X e RETURN. Il programma e' ancora perfettamente in memoria, e anche il valore delle variabili e' conservato.
. Fai attenzione a non premere RUN/STOP mentre stai accendendo il calcolatore, altrimenti entri in MONITOR, ma uscendo da esso, con X, il BASIC non e' in grado di funzionare, poiche' in questo modo e' stata saltata la routine di inizializzazione del BASIC.

1.7 EDITOR

Se la lettura di questo paragrafo ti risulta difficile, ritorna a considerarlo piu' avanti, dopo che avrai fatto un po' di esperienza.

Viene chiamato EDITOR quella parte di SISTEMA OPERATIVO che provvede a gestire il colloquio tra la tastiera e il video. L'EDITOR del tuo COMMODORE 16 ha delle particolarita' che e' bene mettere a fuoco.
Innanzi tutto e' un EDITOR FULL SCREEN. Questo vuol dire che quando premi RETURN viene considerata valida tutta la linea dove si trova il cursore dal precedente RETURN all'attuale RETURN, e non solo i caratteri premuti ex novo. Questa particolarita' aiuta molto durante la fase di correzione dei programmi: non occorre digitare tutta la linea sbagliata, ma basta

correggere solo la parte sbagliata, e premere RETURN.
Inoltre accetta delle linee lunghe fino a 88 caratteri, poiche' considera linee logiche anziche' linee fisiche. Il fatto che linee piu' lunghe di 88 caratteri diano il messaggio STRING TOO LONG e' dovuto alla dimensione limitata della zona di memoria dedicata ad accettare i caratteri in ingresso dalla tastiera, e non a limiti imposti dall'EDITOR.

LINEE LOGICHE E LINEE FISICHE

Sul video del tuo televisore, appena accendi il calcolatore, appare uno schermo dove puoi scrivere dei caratteri, digitandoli dalla tastiera. Se conti i caratteri che puoi porre su una riga, vedi che sono 40; se poi conti quante righe e' possibile scrivere sullo schermo, vedi che sono 25.
Le righe che abbiamo appena contato, le chiamiamo LINEE FISICHE; esse infatti sono fisicamente 25, e il loro numero non puo' variare, ne' la loro lunghezza puo' superare i 40 caratteri. Abbiamo accennato che il calcolatore accetta in ingresso linee fino a un massimo di 88 caratteri. E' lecito domandarsi come sia possibile creare una linea di 88 caratteri su uno schermo che possiede solamente 40 colonne. La spiegazione e' che le linee in questione non sono linee fisiche (tali linee infatti contengono sempre 40 caratteri, di cui alcuni eventualmente sono spazi bianchi), ma LINEE LOGICHE. Il concetto di linea logica e' molto semplice; una LINEA LOGICA e' un insieme di una o piu' linee fisiche. Una linea logica inizia dopo la pressione del tasto RETURN, e continua finche' non si preme nuovamente RETURN, a patto di non usare i tasti di controllo del cursore e CLEAR/HOME. Se viene premuto un tasto dopo il quarantesimo della linea corrente, l'EDITOR porta il cursore nella nuova linea, e "collega" la nuova linea a quella precedente. Lo stesso

avviene dopo l'ottantesimo carattere. Esiste nell'area di memoria riservata al SISTEMA OPERATIVO una "mappa", che tiene conto delle linee fisiche che sono l'inizio di una linea logica, e di quelle che invece sono il proseguimento della linea logica precedente.

I "MODI" DELL'EDITOR: "QUOTE" E "INSERT"

Se hai gia' avuto modo di scrivere qualche programma sul COMMODORE 16, hai notato probabilmente che accade qualcosa di strano quando premi i tasti SHIFT-2 (cioe' apri le virgolette); alcuni dei caratteri di controllo vengono stampati in maniera strana, anziche' essere immediatamente eseguiti. Avviene che l'EDITOR entra nel modo QUOTE (modo virgolette). Quando l'EDITOR si trova in tale stato, i caratteri di controllo, escluso RETURN, vengono stampati come particolari caratteri in campo inverso, dentro le virgolette. Questi caratteri di controllo verranno eseguiti quando la STRINGA (insieme di caratteri) sara' stampata. Come esempio digita:

A$="SHIFT-CLEAR/HOME"

e premi RETURN.

Il tasto SHIFT-CLEAR/HOME appare come un cuoricino in campo inverso. Digita poi:

PRINT A$

e premi RETURN.

Vedi che tutto lo schermo e' stato cancellato, e il cursore si trova nell'angolo in alto a sinistra. E' stato "stampato" il carattere di controllo CLEAR (cancella lo schermo) e lo schermo e' stato cancellato. L'utilita' dell'uso dei caratteri di controllo all'interno di stringhe da stampare e' evidente. Puoi controllare da programma la posizione del cursore, ed eseguire i caratteri di controllo come FLASH ON/OFF, REVERSE ON/OFF, i tasti per cambiare il colore del cursore, ecc.

Per uscire dal modo QUOTE esistono diversi modi:

. Premere RETURN e terminare la linea.

. Premere SHIFT-RETURN e portare il cursore al-

la nuova linea, senza introdurre la linea corrente.
. Chiudere le virgolette.
. Premere ESC O (non zero, ma O come Otranto. Vedi Paragrafo 1.4).
Il modo INSERT (modo inserimento) e' fondamentalmente identico al QUOTE, ma diverso e' il modo di entrarvi; si entra in INSERT premendo i tasti SHIF-INST/DEL, e vi si rimane finche' non si sono premuti tanti tasti quanti erano gli spazi inseriti. Ovviamente si puo' uscire dal modo INSERT anche nei modi con cui si esce dal modo QUOTE (escluso chiudere le virgolette).

1.8 RIEPILOGO

In questo capitolo si e' parlato di:

.TASTI NUMERICI
.TASTI ALFABETICI
.TASTI CARATTERI VARI
.TASTI GRAFICI
.TASTI FUNZIONE
.TASTI DI CONTROLLO
.TASTI DI SERVIZIO
.EDITOR
.LINEE LOGICHE
.LINEE FISICHE

se non ritieni di avere le idee chiare su qualcuno di questi argomenti, ti conviene far girare di nuovo CAP1 e rileggere questo capitolo.

CAPITOLO 2

IL VIDEO

2.1 CARATTERISTICHE E NOMENCLATURA

Il video e' il piu' immediato dispositivo di output, cioe' di uscita. Come gia' sai, con il COMMODORE 16 sul video puoi scrivere 1000 caratteri: 25 righe di 40 caratteri ciascuna. Nel COMMODORE 16 esistono due insiemi di caratteri raggruppati nel SET MAIUSCOLO/GRAFICO e nel SET MINUSCOLO/MAIUSCOLO. Per passare da un SET all'altro devi premere contemporaneamente i tasti CBM LOGO e SHIFT. Esistono comunque 3 tipi di caratteri:

a) numerici
b) alfabetici
c) grafici.

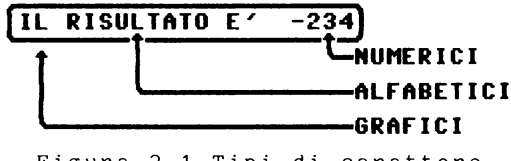


Figura 2.1 Tipi di carattere

I caratteri di tipo grafico possono essere usati per abbellire o rendere piu' leggibili le stampe dei risultati delle elaborazioni del tuo COMMODORE 16 o per cominciare a creare qualche semplice giochino o animazione (come vedrai piu' avanti).
Quando accendi il calcolatore, sul video vedi uno SCHERMO bianco, attorniaazzurro, delle

scritte nere (del set MAIUSCOLO/GRAFICO) e, poco piu' sotto le scritte, un quadratino nero che lampeggia: il CURSORE. Il cursore indica il punto in cui verra' scritto il prossimo carattere e puo' essere spostato grazie ai quattro tasti di movimento del cursore posti in alto a destra sulla tastiera e contraddistinti dalle frecce. Se sposti il cursore su delle scritte gia' esistenti, queste non verranno cancellate, ma il cursore segnalera' la sua presenza alternando, in quella posizione, il carattere stesso e il corrispondente carattere in campo inverso.

2.2 I COLORI E GLI EFFETTI SPECIALI

Il COMMODORE 16 e' un calcolatore che ha la possibilita' di visualizzare a colori: puoi scegliere i colori che preferisci per il bordo, lo sfondo e le scritte. Il modo piu' semplice per cambiare il colore del cursore, e quindi di tutti i caratteri che scriverai da quel momento in poi, e' quello di premere insieme i tasti CTRL e un tasto da 1 a 8 o CBM LOGO e un tasto da 1 a 8. Nel primo caso il cursore diventera' del colore scritto, in inglese, in alto, sulla parte frontale del tasto numerico premuto; nel secondo caso del colore scritto in basso. Premendo CTRL e "," tutti i caratteri, da quel momento in poi, vengono scritti lampeggianti (hai attivato FLASH ON); questo effetto dura fino a che non viene premuto RETURN o CTRL e "." (FLASH OFF).
Premendo CTRL e 9 (che attiva RVS ON), invece, i caratteri verranno scritti in campo inverso fino a che non premi RETURN o CTRL e 0 (RVS OFF).
Scegliendo il colore del cursore con i tasti CTRL e CBM LOGO puoi scegliere tra 16 colori, ma i colori del COMMODORE 16 sono di piu': ben 121! Come fare, quindi, per scegliere tra tutti questi colori? Devi usare il comando COLOR del BASIC.

Se tu scrivi, su una linea dove non esistono altre scritte:
COLOR 1,7,6
e quindi premi RETURN, il COMMODORE 16 ti risponde "READY." in blu.

Il primo numero, che segue il comando COLOR, indica quale colore e' da cambiare:

0 cambia il colore dello schermo
1 cambia il colore del cursore
2 cambia il colore multicolore 1 (Capitolo 7)
3 cambia il colore multicolore 2 (Capitolo 7)
4 cambia il colore del bordo

nel nostro caso il primo numero e' 1, quindi e' stato cambiato il colore del cursore (e la scritta "READY." e' stata scritta col nuovo colore).
Il secondo numero indica il colore scelto:

1	nero	9	arancione
2	bianco	10	marrone
3	rosso	11	giallo-verde
4	ciano	12	rosa
5	viola	13	verde-blu
6	verde	14	blu chiaro
7	blu	15	blu scuro
8	giallo	16	verde chiaro

nel nostro caso il secondo numero e' 7, quindi il colore scelto e' il blu.
L'ultimo numero indica la luminosita' del colore. Puoi scegliere un numero tra 0 (scurissimo) e 7 (chiarissimo); nel nostro esempio abbiamo scelto 6 cioe' molto chiaro.
Nota che il colore blu e il colore blu chiaro o il verde e il verde chiaro, anche se usati con la stessa luminosita', non sono identici. I colori disponibili sono quindi 15 (dal 2 al 16) X 8 (le luminosita') piu' il nero (per il quale non valgono le luminosita'); in tutto dunque 121!

2.3 RIEPILOGO

In questo capitolo si e' parlato di:

.CARATTERI
.COLORI
.EFFETTI FLASH e REVERSE
.LUMINOSITA'
.CAMBIAMENTO COLORI: SFONDO, BORDO, CARATTERI

se non ti senti sicuro su questi argomenti, fai girare di nuovo il programma CAP2, e poi rileggi questo capitolo.

CAPITOLO 3

CALCOLATRICE

3.1 IL COMMODORE 16 COME CALCOLATRICE

Come hai visto seguendo il programma CAP3, puoi usare il tuo COMMODORE 16 per eseguire calcoli anche complicati.
Quando usi il calcolatore in questo modo, ti servi del linguaggio BASIC in modo IMMEDIATO, cioe' scrivi una istruzione e la esegui quando premi il tasto RETURN. Le istruzioni che abbiamo visto sono:
. PRINT, che puo' essere scritta anche come ?, cioe' il calcolatore interpreta il ? come se tu avessi scritto la parola PRINT; questa istruzione fa scrivere sul video quello che segue, se segue un'espressione, viene scritto il risultato del calcolo.
. ISTRUZIONE DI ASSEGNAZIONE, che si scrive:
  nome-variabile=espressione.
Essa assegna alla variabile il valore dell'espressione; se la variabile non esiste gia' viene creata. Il valore assegnato alla variabile rimane immutato fino a quando si esegue un'altra istruzione di assegnazione usando lo stesso nome, oppure si impartisce una istruzione che azzera tutte le variabili (come CLR).
Dopo aver assegnato un valore ad una variabile questa puo' essere utilizzata in qualunque espressione.

Le ESPRESSIONI sono formate da costanti (numeri), da variabili, da funzioni, da parentesi rotonde e da operatori aritmetici. Esse devono

essere scritte utilizzando le regole della matematica.

Gli OPERATORI ARITMETICI sono:

^ elevamento a potenza	- segno
* moltiplicazione	/ divisione
+ addizione	- sottrazione.

Gli elementi costanti, variabili e funzioni, devono essere collegati tra loro mediante gli operatori aritmetici e le parentesi rotonde.
Non si possono avere due operatori aritmetici vicini, salvo nel caso del segno -. Si puo' scrivere: 6*-8
ma questo risulta poco chiaro, per cui e' meglio scrivere: 6*(-8).
Non si puo' scrivere 6*/8, perche' non ha senso usare i due operatori * e / vicini.

Le VARIABILI sono dei CONTENITORI di dati; ad esse vengono assegnati dei NOMI secondo regole precise.
Il nome di una variabile puo' essere formato da due caratteri, dei quali il primo deve essere una lettera e il secondo puo' essere una lettera o una cifra numerica e puo' anche mancare.

Negli esempi di calcolo visti sul video abbiamo sempre parlato di numeri in generale; in realta' il COMMODORE 16 puo' trattare due tipi di numeri:
. i NUMERI INTERI,
. i NUMERI REALI (quelli non interi).
Per distinguere il nome di una variabile che puo' contenere solo numeri interi da quello di una variabile che puo' contenere numeri reali, si fa seguire al nome il suffisso %. La variabile A% contiene un numero intero, si chiama anche variabile intera. La variabile A contiene un numero reale, si chiama anche variabile reale.
Se in una operazione di assegnazione a sinistra dell'uguale compare una variabile intera, il

risultato dell'espressione viene reso intero prima di assegnarlo alla variabile. Per esempio:
B%=3.5, assegna a B% il valore intero 3.

I numeri interi purtroppo sono molto limitati in grandezza, cioe' una variabile intera puo' contenere un numero compreso tra -32768 e +32767. Se si cerca di assegnarle un numero fuori da questo intervallo, esso viene troncato perdendo il suo valore e ovviamente i risultati dei calcoli sono errati.
Nel calcolatore viene usata una rappresentazione approssimata per i numeri reali (cioe' con un numero limitato di cifre decimali dopo il punto decimale).
Nel COMMODORE 16 i numeri reali sono conservati con circa 10 cifre significative; il valore varia tra -4294967304 e +4294967304.
Essi possono essere rappresentati in uscita in due modi:
. in VIRGOLA FISSA,
. in VIRGOLA MOBILE (FLOATING-POINT) o FORMA ESPONENZIALE.
Il COMMODORE 16 decide di evidenziare un numero in uno dei due formati in base al numero delle cifre significative presenti. Fino a 9 cifre, ed escludendo un piccolo intervallo intorno allo zero (precisamente i numeri X che risultano: -0.01<X<0 oppure 0<X<0.01), il numero viene evidenziato in virgola fissa; gli altri numeri vengono evidenziati in virgola mobile. Nella Figura 3.1 sono indicati gli intervalli validi per le due rappresentazioni.

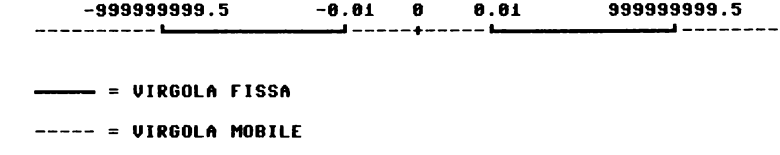


Figura 3,1 Intervalli rappresentazioni numeri

Il ricorso alla rappresentazione in virgola mobile consente di limitare il numero di caratteri.
Infatti se possiamo scrivere un numero in una casella di un foglio di carta formata da 20 caselline, ognuna capace di registrare una cifra, non possiamo scrivere piu' di 20 cifre.
In tale caso un numero come:
0.00000000000000123456789123456 78 non sappiamo come scriverlo; infatti se scriviamo le prime 20 cifre abbiamo una buona approssimazione sulla grandezza del numero, ma perdiamo un bel gruppo di cifre significative, mentre se scriviamo le ultime 20 cifre cambiamo completamente la natura del numero, che diventa grande.
Analogamente un numero come:
12345678912345678900000000000, scritto in 20 cifre cambia completamente ordine di grandezza.

Se decidiamo di usare le 20 caselline a disposizione in questo modo:
. le prime 3 per il segno e l'esponente da dare a 10 per rappresentare l'ordine di grandezza del numero (+xx o -xx); arriviamo a ordini di grandezza da 10 elevato a -99 (piccolissimo) a 10 elevato a +99 (molto grande),
. le altre 17 per il segno e le cifre significative del numero,
per ottenere il valore del numero moltiplichiamo le 17 cifre conservate (16 piu' il segno) per l'opportuna potenza di 10 (10^esponente). In questo caso conserviamo l'ordine di grandezza del numero e il maggior numero possibile di cifre significative. Inoltre e' inutile visualizzare gli zeri a sinistra o a destra in quanto ne puo' tener conto l'esponente.
Da quanto detto viene giustificata la terminologia "in virgola mobile"; infatti la virgola viene spostata dopo la prima cifra significativa del numero, come se tutti i numeri fossero del tipo: x.xx...x; viene tenuto conto

degli spostamenti della virgola (che e' il punto decimale) nell'esponente.

Per i due numeri visti sopra:

0.00000000000000012345678912345678
prendendo solo 17 cifre significative:
+1.2345678912345678 * 10^-15

12345678912345678900000000000
prendendo solo 17 cifre:
+1.2345678912345678 * 10^+27

Il COMMODORE 16 fa proprio un lavoro di questo tipo per conservare i numeri reali nella sua memoria, cioe' li conserva sempre in virgola mobile.
Il numero di cifre consentite e' diverso da quello citato nell'esempio, e, inoltre, il calcolatore lavora al suo interno con l'aritmetica binaria. Nell'aritmetica binaria sono usate solo due cifre, 0 e 1, e il valore posizionale delle cifre viene calcolato in base alle potenze di 2. Tu non hai bisogno di occuparti dell'aritmetica binaria, infatti il COMMODORE 16 comunica con l'esterno con la normale aritmetica decimale.

In uscita il COMMODORE 16 ci mostra i numeri reali in virgola fissa se hanno fino a 9 cifre (escludendo gli intervalli intorno allo zero sopra citati) e in virgola mobile negli altri casi. Per rendere evidente quest'ultimo tipo di rappresentazione il numero viene mostrato nella forma:

```
 X.XX...XE+YY      X.XX...XE-YY
-X.XX...XE+YY     -X.XX...XE-YY
```

e il valore di YY puo' essere al massimo 38, mentre le cifre X...X possono essere al massimo 9.

Le FUNZIONI corrispondono a un insieme di operazioni che vengono eseguite citando il nome dalla funzione seguito tra parentesi dall'ARGOMENTO da usare nel calcolo. L'argomento puo' essere una costante, una variabile o una espressione, purche' il valore sia tale da non contraddire la logica della funzione, tipo estrazione della radice quadrata di un numero negativo.

Le OPERAZIONI ARITMETICHE si ottengono usando gli operatori aritmetici. Il simbolo "=" ha il normale significato di uguaglianza usato in matematica, con qualcosa in piu'. Il normale significato di uguaglianza e' che quanto sta a sinistra del simbolo e' uguale a quanto sta a destra. In programmazione il simbolo "=" puo' essere usato in questo modo per indicare una relazione tra due entita', che puo' essere vera o non vera, oppure con il significato di assegnazione, cioe' alla variabile che sta a sinistra del simbolo viene assegnato il valore che risulta dall'espressione che sta a destra, espressione che puo' ridursi a una semplice costante. In questa ottica si puo' scrivere: X=X+1, che dal punto di vista matematico e' un non senso, ma dal punto di vista della programmazione significa "aggiungi 1 al precedente contenuto della variabile X".

Le ESPRESSIONI vengono calcolate partendo da sinistra e muovendosi verso destra, ma dando la precedenza ai calcoli indicati tra parentesi rotonde. Inoltre tra gli operatori aritmetici le precedenze sono:
. elevamenti a potenza,
. assegnazioni del segno meno,
. moltiplicazioni e divisioni,
. somme e sottrazioni.

Se nelle espressioni compaiono delle funzioni, esse sono calcolate subito e viene sostituito ad esse il loro valore. Se l'argomento delle

funzioni e' un'espressione, essa viene calcolata prima della funzione.

3.2 RIEPILOGO

In questo capitolo sono stati trattati i seguenti soggetti:

.MODO DI ESECUZIONE CALCOLI
.ISTRUZIONE PRINT
.ISTRUZIONE DI ASSEGNAZIONE
.ESPRESSIONI
.OPERATORI ARITMETICI
.VARIABILI NUMERICHE INTERE
.VARIABILI NUMERICHE REALI
.NUMERI INTERI
.NUMERI REALI
.RAPPRESENTAZIONE IN VIRGOLA FISSA
.RAPPRESENTAZIONE IN VIRGOLA MOBILE
.FUNZIONI MATEMATICHE
.USO DELLE PARENTESI

se credi di avere dei dubbi su qualcuno di questi argomenti, ti conviene rivedere il programma CAP3, rileggere il Capitolo 3 e fare molte prove sul calcolatore.

CAPITOLO 4

IL PROGRAMMA

4.1 COSA E' UN PROGRAMMA

Un PROGRAMMA per calcolatore e' formato da una sequenza di istruzioni eseguibili da parte del calcolatore.
Ogni calcolatore e' costruito in modo da essere in grado di eseguire in modo automatico un programma scritto nel suo LINGUAGGIO MACCHINA e preventivamente registrato nella sua MEMORIA, secondo prescritte modalita'.
La redazione di programmi in linguaggio macchina e' abbastanza complessa e viene considerata un lavoro per specialisti; non si tratta comunque di un lavoro trascendentale, e, a nostro avviso, chiunque dotato di media intelligenza, pazienza e buona volonta' puo' imparare a programmare un calcolatore in linguaggio macchina.
A partire dall'avvento dei primi calcolatori, nella seconda meta' degli anni 50, e' stato affrontato il problema di mettere a punto linguaggi simbolici di programmazione, adatti a consentire la diffusione della pratica della programmazione e quindi di un utilizzo massivo dei calcolatori. Si trattava di inventare dei linguaggi adatti ad essere appresi dall'uomo, e basati su regole tali da consentirne senza ambiguita' la conversione in linguaggio macchina. Sono stati creati molti tipi diversi di linguaggi, ognuno con pregi e difetti, ma tutti molto utili alla diffusione dei calcolatori.

Nel COMMODORE 16 la memoria e' divisa in due parti: una parte contiene preregistrati dei programmi e viene chiamata memoria ROM (Read Only Memory, memoria a sola lettura); l'altra, che viene chiamata memoria RAM (Random Access Memory, memoria ad accesso casuale, da interpretare come memoria per lettura e scrittura), serve per registrare i programmi e i dati dell'utente. La parte di programmi preregistrata comprende il SISTEMA OPERATIVO e l'INTERPRETE BASIC. Il SISTEMA OPERATIVO comprende tutti i programmi che sovraintendono al funzionamento del calcolatore; esso e' formato da diversi moduli, ognuno con funzioni specifiche, come, per esempio, la gestione delle periferiche. Al momento dell'accensione del calcolatore e' attivo il modulo (EDITOR) che gestisce il colloquio con l'utente in modo strettamente connesso al linguaggio BASIC. Il COMMODORE 16, come moltissimi altri calcolatori personal, e' predisposto alla programmazione in BASIC. E' possibile programmare il calcolatore anche in linguaggio macchina, ma l'accesso al linguaggio macchina si ha sempre partendo dal BASIC.
Dal momento che il calcolatore capisce solo il suo linguaggio macchina, per poter eseguire un programma scritto in un altro linguaggio e' necessario che sia presente nella memoria del calcolatore un programma, scritto in linguaggio macchina, che funga da intermediario e traduca le istruzioni BASIC in istruzioni eseguibili. Tale intermediario e' l'INTERPRETE BASIC.
La scritta che compare all'accensione del calcolatore (vedi Paragrafo 0.2) dice che il COMMODORE 16 contiene l'interprete BASIC nella versione 3.5, e che tu puoi disporre di 12277 byte di memoria RAM per il tuo programma.

Indipendentemente dal linguaggio di programmazione usato e' necessario capire cosa e' un programma per calcolatore. Esso e' una sequenza

di istruzioni elementari che descrivono, passo dopo passo, cosa il calcolatore deve fare, in modo che:
. partendo da dati iniziali,
. svolga un certo tipo di trasformazione dei dati,
. produca dei risultati.
In sostanza un programma opera una trasformazione di dati, intendendo questo concetto in modo molto generale. Possiamo considerare una trasformazione di dati cose come:
. risolvere un'equazione di secondo grado,
. calcolare la lunghezza della circonferenza di un cerchio,
. calcolare un volume,
. elaborare e stampare una fattura,
. mettere in ordine alfabetico dei nominativi,
. ricercare quante volte un determinato aggettivo compare in un testo,
. giocare a scacchi,
e quante altre se ne vogliano aggiungere.
Qualunque programma tu voglia scrivere, la prima cosa da fare e' conoscere bene il problema che il programma deve affrontare e risolvere; questo e' di gran lunga il compito piu' difficile per il programmatore.

4.2 IL LINGUAGGIO BASIC

Il linguaggio BASIC e' di tipo:
. SIMBOLICO,
. INTERPRETATIVO,
. CONVERSAZIONALE.

SIMBOLICO, perche' lavora su simboli facilmente comprensibili, che possono essere parole chiave del linguaggio e nomi simbolici creati dal programmatore.
INTERPRETATIVO, perche' durante l'esecuzione del programma e' presente nella memoria del calcolatore il programma INTERPRETE BASIC, che prov-

vede alla interpretazione, traduzione e esecuzione delle frasi del linguaggio.
CONVERSAZIONALE, perche' si svolge un colloquio tra calcolatore e utente, che rende molto semplice la stesura, la correzione e l'esecuzione del programma.

Possiamo schematizzare questo grande e potente foglio elettronico, che e' la memoria del nostro COMMODORE 16, immaginandola divisa nelle seguenti parti:
. ROM SISTEMA OPERATIVO,
. ROM INTERPRETE BASIC,
. RAM programma utente in BASIC,
. RAM dati del programma, del SISTEMA OPERATIVO e dell'INTERPRETE BASIC,
come indicato in Figura 4.1.

Figura 4.1 Schema della memoria

Esistono molte versioni del linguaggio BASIC; e' comune a tutte la stessa filosofia di gestione

delle risorse del calcolatore, ma possono esserci anche molte differenze, tali da rendere una versione molto piu' potente di un'altra.
Il COMMODORE 16 e' dotato della versione 3.5 del BASIC, derivata dalla versione originale della MICROSOFT, con aggiunta di nuovi comandi, che la rendono molto potente. E' molto interessante avere su un personal di basso costo una cosi' ricca versione del BASIC, come questa marcata 3.5.

4.3 COME SI ORGANIZZA UN PROGRAMMA

Un programma va organizzato procedendo nello studio del problema che si vuole risolvere.
Il primo passo e' la definizione esatta e completa del problema che deve essere affrontato. In generale, dopo una prima definizione e l'inizio della relativa riflessione, il problema puo' anche essere ridefinito in modo piu' opportuno.
A questo punto e' necessario organizzare in modo preciso i dati sui quali il programma deve lavorare. Dei dati interessano diversi aspetti:
. la loro natura,
. la loro quantita',
. la loro fonte.
Si deve arrivare alla stesura di uno schema contenente tutte le caratteristiche dei dati di ingresso, chiamati dati di INPUT.
Esaurito l'argomento dei dati di INPUT, deve essere affrontato quello dei risultati, cioe' dei dati di OUTPUT:
. cosa si vuole ottenere,
. in quale forma,
. su quale mezzo.
Nel caso di una stampa su carta, deve, per esempio, essere precisato cosa scrivere su ogni riga, a quale distanza deve trovarsi ogni dato dal precedente, e cosi' via.

Avendo chiarito da dove si parte e dove si vuole arrivare, e' necessario stabilire come arrivarci. Si tratta cioe' di scegliere la procedura da seguire; si dice anche scegliere gli algoritmi piu' adatti. Con la parola ALGORITMO si intende "modo di procedere", cioe' sequenza di operazioni, necessarie per operare la desiderata trasformazione di dati.
Abbiamo parlato di operazioni; esse possono essere di tipo matematico o logico, e con esse si puo' costruire qualunque programma, anche molto complesso.

Lo studio del problema comporta la stesura di note, che possono essere organizzate in modi diversi. Alcuni preferiscono degli appunti schematici, nei quali i successivi passi da compiere ricevono una numerazione progressiva. Altri preferiscono ricorrere alla stesura di diagrammi, cioe' di schemi grafici, formati da blocchetti di diversa forma, ognuno con un particolare significato, all'interno dei quali sono scritte note sintetiche esplicative.
In generale si consiglia un metodo strutturato, nel quale, partendo da una definizione molto generale del problema, si scende, per affinamenti successivi, alla stesura finale, che assomiglia gia' molto alla CODIFICA (scrittura) del programma nel linguaggio di programmazione scelto.

Tutto il lavoro che deve essere svolto prima di arrivare alla codifica di un programma va sotto il nome di: ANALISI DEL PROBLEMA.

Ti potrai chiedere se quando inizi lo studio di un problema devi decidere a priori in quale linguaggio verra' codificato il programma. La risposta e' che questo non e' a priori necessario, anche se la scelta del linguaggio di programmazione influisce sul modo di immissione ed emissione dei dati, e, spesso, anche sulla elaborazione di essi. E' comunque certo che, se

un problema e' stato ben studiato, esso puo' essere tradotto facilmente in un programma scritto in un qualunque linguaggio.

I dati che vengono trattati da un programma possono essere costanti o variabili. I dati costanti possono essere introdotti direttamente nelle linee di programma come numeri interi o decimali (ricorda il Capitolo 3), scritti nel formato a virgola fissa o a virgola mobile, o come sequenze di caratteri alfanumerici, che vengono chiamate STRINGHE. I dati, che variano durante l'esecuzione del programma, come quelli che si ricevono in INPUT, i risultati intermedi o finali, costituiscono le VARIABILI del programma. Ogni VARIABILE viene definita assegnandole un nome simbolico, che rispetti le regole del linguaggio.
Il dato viene richiamato citando il nome della variabile che lo contiene.

4.4 LE ISTRUZIONI DEL BASIC

Le istruzioni del BASIC sono formate da PAROLE CHIAVE del linguaggio e da PARAMETRI. Con PAROLE CHIAVE si intendono una o piu' parole che hanno un particolare significato operativo. Con PARAMETRI si intende tutto quello che deve essere scritto oltre le parole chiave per rendere completa un'istruzione.
Le istruzioni del linguaggio sono organizzate in linee che possono comprendere piu' di una istruzione. Se una linea comprende piu' istruzioni, esse devono essere scritte separandole con il carattere separatore "due punti" (:). Il carattere ":" ha per il BASIC il significato di separatore.
Una linea di istruzioni puo' essere scritta ponendo inizialmente un numero: NUMERO DI LINEA, oppure iniziando subito con la prima istruzione. Questo diverso modo di scrivere una linea

da' luogo a un comportamento diverso al momento della pressione del tasto RETURN, che chiude la linea. Se la linea inizia con il NUMERO DI LINEA, essa viene memorizzata nella zona di memoria dedicata al programma utente, assegnandole la giusta posizione in base al numero di linea. Le linee di istruzioni vengono memorizzate per numero di linea crescente. Se la linea non inizia con un numero essa viene eseguita immediatamente al momento della pressione del tasto RETURN.

Il BASIC puo' dunque lavorare in due modi:
. MODO IMMEDIATO (linee senza numero),
. MODO DIFFERITO (linee numerate).
Vedremo procedendo che non tutte le istruzioni possono essere eseguite nei due modi.
Abbiamo gia' fatto esperienza del modo di esecuzione immediato nel Capitolo 3.

Nell'Appendice A e' riportata la scheda del BASIC 3.5 del COMMODRE 16 e ad essa ti rimandiamo per una visione completa del linguaggio.
Raggruppiamo le istruzioni in classi, per avere uno schema sintetico delle possibilita' del linguaggio.
Abbiamo:
..Istruzioni per l'ingresso dei dati (operazioni di INPUT): DATA, GET, GET#, GETKEY, INPUT, INPUT#, JOY, READ, RESTORE.

..Istruzioni per l'uscita dei dati (operazioni di OUTPUT): CHAR, POS, PRINT, PRINT#, USING, PUDEF, SPC, TAB.
..Istruzioni e funzioni di calcolo: ABS, ATN, COS, DEC, EXP, INT, LET, LOG, RND, SGN, SIN, SQR, TAN, VAL.
..Istruzioni e funzioni per dati alfanumerici: ASC, CHR$, HEX$, INSTR, LEFT$, LEN, MID$, RIGHT$, STR$.
..Istruzioni di controllo: DO...LOOP (UNTIL, WHILE, EXIT), GOTO, GOSUB, IF...THEN...ELSE,

FOR...TO...STEP, NEXT, ON...GOTO, ON...GOSUB, RETURN.
..Istruzioni per grafica ad alta risoluzione e multicolore: BOX, CIRCLE, COLOR, DRAW, GRAPHIC, GSHAPE, LOCATE, PAINT, RCLR, RDOT, SCALE, SCNCLR, SSHAPE.
..Istruzione per suonare: SOUND, VOL.
..Istruzioni di servizio per la stesura e la gestione del programma e dei dati: AUTO, BACKUP, CLOSE, CLR, CMD, COLLECT, CONT, COPY, DEF FN, DELETE, DIM, DIRECTORY, DLOAD, DSAVE, END, FRE, HEADER, KEY, LIST, LOAD, MONITOR, NEW, OPEN, PEEK, POKE, REM, RENAME, RENUMBER, RESUME, RGR, RLUM, RUN, SAVE, SCRATCH, STOP, SYS, TRAP, TROFF, TRON, USR, VERIFY, WAIT.

Le regole per la formazione dei nomi delle variabili sono le seguenti:
. Il nome deve essere formato da al massimo due caratteri (ne puoi usare di piu', ma vengono riconosciuti solo i primi due), il primo deve essere una lettera, il secondo puo' essere una lettera o una cifra.
. Il nome deve essere seguito dal suffisso % per indicare numeri interi.
. Il nome deve essere seguito dal suffisso $ per indicare variabili stringa.
. I nomi senza suffisso si riferiscono a variabili reali. Le variabili sopra descritte sono VARIABILI SINGOLE, ogni nome corrisponde a un dato.

4.5 COME SI SCRIVE UN PROGRAMMA

Quando accendi il COMMODORE 16 esso e' gia' pronto e aspetta che tu scriva un programma o impartisca istruzioni da eseguire in modo immediato.
E' attivo l'EDITOR, che ti consente l'uso del video e della tastiera (vedi Paragrafo 1.7).
Se vuoi essere sicuro che la zona di memoria de-

dicata al programma sia pulita, devi scrivere: NEW e premere RETURN.
Per effetto di questo comando, impartito qui in modo immediato, la memoria viene pulita dal programma e dai dati eventualmente presenti.
Se desideri vedere un video pulito e cominciare a scrivere in alto, puoi premere i tasti SHIFT-CLEAR/HOME.
A questo punto puoi scrivere le tue linee di istruzioni numerandole progressivamente. Se ti sbagli e dimentichi una linea, e' semplice rimediare, basta scrivere la linea dimenticata iniziando con un numero tale che le consenta di essere posizionata dove desideri.
Mentre procedi nella scrittura delle linee, il video si riempie e puo' aver luogo lo scorrimento delle scritte verso l'alto (scrolling).
Se ti accorgi che qualche linea presente sullo schermo e' errata puoi riscriverla ex novo o spostare il cursore, per mezzo dei relativi tasti, posizionarti dove desideri e correggere; la linea modificata, o riscritta, va a sostituirsi alla precedente al momento della pressione del tasto RETURN.
Se desideri abolire una linea basta scriverne il numero e premere RETURN.
Se desideri rivedere una parte di programma che e' scomparsa dal video per effetto dello scrolling, puoi chiederne la lista con il comando LIST. Solo che se scrivi LIST e premi RETURN e il programma che sta in memoria ha piu' di 25 linee perdi le prime. Puoi usare una forma diversa del comando LIST; scrivere per esempio: LIST -100, e ottieni la lista delle linee di programma fino a quella numerata 100. Se invece scrivi: LIST 40-120, ottieni la lista delle linee numerate da 40 a 120. Se scrivi: LIST 80, ottieni la lista della sola linea 80. Se scrivi: LIST 110-, ottieni la lista delle linee dala 110 in avanti.
Il tuo programma rimane in memoria fino a quando non scrivi NEW oppure togli corrente al COMMODORE 16.

4.6 LE ISTRUZIONI PIU' ELEMENTARI DEL BASIC

Ci proponiamo di risolvere il seguente problema:
CHIEDERE UN NUMERO, CALCOLARNE IL QUADRATO E IL CUBO E MOSTRARE I RISULTATI.

Per avere una certa liberta' dobbiamo lavorare con numeri reali, infatti la grandezza consentita per i numeri interi ci bloccherebbe subito. Dobbiamo scegliere un nome per la variabile che deve ricevere da tastiera il numero; la chiamiamo N. Il numero N e' l'unico dato di INPUT necessario per risolvere il nostro problema. Dobbiamo poi creare due nomi di variabili per contenere rispettivamente il quadrato e il cubo di N; scegliamo N2 per il quadrato e N3 per il cubo. L'algoritmo di calcolo e' molto semplice, infatti:
N2=N*N e N3=N2*N. Abbiamo gia' visto nel Capitolo 3 che l'asterisco si usa come operatore per la moltiplicazione. Le formule di calcolo possono anche essere scritte in un altro modo:
N2=N^2 e N3=N^3, dove la freccia verso l'alto significa "elevato a".
Decidiamo di fornire i risultati in modo chiaro, scrivendo sul video:
N=...
IL QUADRATO DI N E' UGUALE A ...
IL CUBO DI N E' UGUALE A ...
Descriviamo verbalmente la procedura da programmare:
.1) legge dalla tastiera un numero N;
.2) calcola il quadrato di N e lo pone in N2;
.3) calcola il cubo di N e lo pone in N3;
.4) stampa i risultati.

Possiamo descrivere la procedura anche con lo schema riportato nella Figura 4.2, che prende il nome di DIAGRAMMA A BLOCCHI.

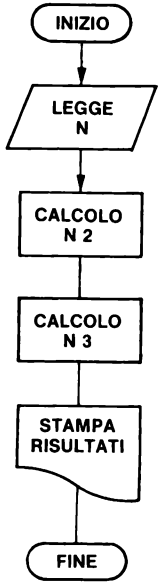


Figura 4.2 Diagramma a blocchi ES4.1

Riportiamo ora il listato del programma BASIC ES4.1 che risolve il problema posto.

```
1 REM ES4.1
10 INPUT"SCRIVI UN NUMERO ";N
20 N2=N↑2:N3=N↑3
30 PRINT"N = ";N
40 PRINT"IL QUADRATO DI N E' UGUALE A: ";N2
50 PRINT"IL CUBO DI N E' UGUALE A: ";N3
```

Ti facciamo osservare i limiti di questo programma; esso lavora per un solo numero, dopo aver eseguito i calcoli e stampato i risultati si ferma.
Commentiamo ora le linee del programma ES4.1.
. 1: inizia con la parola chiave REM, abbrevia-

zione di REMark, che in italiano significa ANNOTAZIONI. Quello che segue REM viene considerato un commento; noi abbiamo scritto il nome attribuito al programma, ma potevamo scrivere qualunque altra cosa. I commenti possono contenere anche il carattere ":", che in questo caso non viene considerato come carattere separatore tra istruzioni; in conseguenza l'istruzione REM deve essere l'ultima di una linea che contenga piu' istruzioni.

. 10: e' l'istruzione per chiedere dati dalla tastiera; alla parola chiave INPUT puo' seguire, come nel nostro caso, un messaggio esplicativo della richiesta di dati, compreso tra virgolette, seguito da ";" e dalla lista delle variabili nelle quali si vuole siano memorizzati i dati letti dalla tastiera, separate da virgola. Il messaggio deve essere dato sotto forma di costante; non si puo' usare il nome di una variabile che lo contenga.

Il calcolatore evidenzia un "?" di richiesta dati (dopo il messaggio, se presente). Devi rispondere con tanti dati quanti sono quelli richiesti, ponendo una virgola alla fine del dato, se ne seguono altri, e premendo RETURN alla fine. Se rispondi con meno dati di quelli richiesti, il calcolatore va a capo e ti mostra "??" a segnalarti che non hai esaurito la richiesta. Devi cercare di rispondere con dati che concordino con il tipo delle variabili presenti nella lista; in caso contrario il calcolatore scrive "?REDO FROM START" e ti chiede nuovamente i dati. Hai questa segnalazione di errore se rispondi con una parola alla richiesta del numero nel nostro programma.

. 20: viene assegnato alla variabile N2 il quadrato del numero N e alla variabile N3 il cubo del numero N. La linea comprende due istruzioni di assegnazione separate da ":". L'istruzione di assegnazione puo' iniziare con la parola chiave LET, cosi': LET N2=N^2, che pero' e' facoltativa. Dopo L'esecuzione dell'istruzione, la

variabile posta a sinistra di "=" contiene il risultato dell'espressione posta a destra.
. 30: la parola chiave PRINT ordina di stampare quello che segue. La lista di stampa puo' comprendere variabili e costanti, che possono essere separate da:
";", per stampare i dati nel loro formato senza aggiunta di spazi intermedi;
",", per stampare il dato passando alla prossima zona di stampa per il successivo. Sul video le zone di stampa sono 4 di 10 caratteri ciascuna.
Noi stampiamo un messaggio e il numero N.
Se la lista di stampa termina senza punteggiatura (, o ;), come nel nostro caso, si ha il passaggio a nuova linea, dopo la stampa (va a capo).
. 40: stampa un messaggio e il numero N2.
. 50: stampa un messaggio e il numero N3. I due risultati appariranno nel formato necessario per il numero di cifre che li compongono.

FORMATI DI STAMPA
I formati di stampa sono:

. uno spazio o il segno meno, il numero, il salto di una posizione, per i dati numerici;
. i caratteri che le compongono per le stringhe.

Riportiamo ora il listato del programma ES4.2, che rappresenta una modifica di ES4.1. In esso abbiamo introdotto dei caratteri di controllo nei messaggi, e precisamente:
. 10: il carattere FLASH ON all'inizio del messaggio e il carattere FLASH OFF alla fine.
. 30: il carattere RVS ON all'inizio del messaggio e il carattere RVS OFF alla fine.
. 40: il carattere CTRL-3, per passare al colore rosso e il carattere RVS ON all'inizio del messaggio, il carattere RVS OFF alla fine. In questo modo e' rimasto attivo il colore rosso anche per la stampa del numero.

. 50: i caratteri RVS ON e RVS OFF all'inizio e alla fine del messaggio.
. 60: abbiamo aggiunto PRINT del carattere CTRL-1 per tornare al colore nero.

```
1 REM ES4.2
10 INPUT"SCRIVI UN NUMERO ";N
20 N2=N↑2:N3=N↑3
30 PRINT"N= ";N
40 PRINT"IL QUADRATO DI N E' UGUALE A: ";N2
50 PRINT"IL CUBO DI N E' UGUALE A: ";N3
60 PRINT"
```

Segue il listato del programma ES4.3, nel quale, rispetto ad ES4.1, abbiamo solo aggiunto la linea 60. Essa contiene una PRINT senza lista per andare a capo e l'istruzione GOTO10. L'istruzione GOTO (che puo' anche essere scritta GO TO), seguita dal numero di una linea del programma, provoca un salto incondizionato al numero di linea citato, cioe' fa continuare l'esecuzione del programma da quel punto.

```
1 REM ES4.3
10 INPUT"SCRIVI UN NUMERO ";N
20 N2=N↑2:N3=N↑3
30 PRINT"N = ";N
40 PRINT"IL QUADRATO DI N E' UGUALE A: ";N2
50 PRINT"IL CUBO DI N E' UGUALE A: ";N3
60 PRINT:GOTO10
```

Il programma ES4.3 e' un programma che non finisce mai. E' un po' laborioso interromperlo, infatti durante la richiesta di dati il calcolatore non riconosce lo STOP; puoi premere contemporaneamente RUN/STOP e il tasto laterale di RESET, per interrompere il programma senza cancellarlo dalla memoria, e poi X e RETURN.

Segue il programma ES4.4, che consiste nella stampa di costanti, numeri e stringhe, per farti comprendere bene come viene gestita la stampa sul video a seconda dei separatori usati nella lista di stampa.

```
1 REM ES4.4
10 S$="12345678901234567890123456789012345678 90"
15 PRINTS$
20 PRINT"ABC","DEF","GHI"
25 PRINTS$
30 PRINT"ABC";"DEF";"GHI"
35 PRINTS$
40 PRINT35,-987,1237
45 PRINTS$
50 PRINT"AABBCCDDEEFFGGHHLLM","NUOVA"
55 PRINTS$
60 PRINT"AABBCCDDEEFFGGHHLLMM","NUOVA"
65 PRINTS$
70 PRINT2345;9876;-5432
```

In ES4.4 la stringa S$, che viene stampata prima di ogni lista di dati, serve per numerare le colonne del video; risulta cosi' piu' facile controllare l'effetto della lista di stampa successiva. Riportiamo i risultati di ES4.4.

```
12345678901234567890123456789012345678 90
ABC        DEF        GHI
12345678901234567890123456789012345678 90
ABCDEFGHI
12345678901234567890123456789012345678 90
 35        -987        1237
12345678901234567890123456789012345678 90
AABBCCDDEEFFGGHHLLM NUOVA
12345678901234567890123456789012345678 90
AABBCCDDEEFFGGHHLLMM          NUOVA
12345678901234567890123456789012345678 90
 2345  9876 -5432
```

Come puoi osservare sono verificate tutte le cose dette fino ad ora, e cioe' l'effetto della virgola e del punto e virgola.

Nel programma ES4.5, richiediamo 3 dati con una istruzione di INPUT. Puoi provare a rispondere premendo RETURN dopo ogni dato per vedere comparire "??".

```
1 REM ES4.5
10 INPUT"SCRIVI TRE NUMERI ";X1,X2,X3
20 N=(ABS(X1*X2-X3))↑(1/2)
25 M=SQR(ABS(X1*X2-X3)):Z=N-M
30 PRINT"X1="X1"X2="X2"X3="X3
40 PRINT"ABS(X1*X2-X3)↑(1/2)="N
50 PRINT"SQR(ABS(X1*X2-X3))="M
60 PRINT"N-M="Z
```

Riportiamo il commento a ES4.5.
. 10: richiesta dei 3 numeri X1,X2 e X3.
. 20: calcolo di N. Nell'espressione abbiamo usato ABS per evitare di elevare a esponente 1/2 (estrazione di radice) un numero negativo.
. 25: calcolo di M. Nell'espressione usiamo la funzione SQR per estrarre la radice, cioe' scriviamo in altro modo lo stesso calcolo di prima. Inoltre calcoliamo in Z la differenza tra N e M.
. 30: stampa i 3 dati di INPUT, preceduti ognuno da un messaggio, senza usare separatori nella lista di stampa.
. 40,50,60: stampa i risultati senza usare separatori nella lista di stampa.

Ricorda che se nelle istruzioni INPUT rispondi solo con RETURN alla richiesta di un dato, la variabile interessata mantiene il suo precedente valore. Inoltre, se prima di scrivere il dato premi la barra di spazio, questi spazi non vengono conservati. Per conservare spazi o altri caratteri, tipo i separatori, all'interno delle

variabili stringa, devi iniziare aprendo le virgolette, e terminare chiudendole.

I programmi esempio di questo paragrafo non sono riportati sulla cassetta allegata, sia perche' sono molto corti, sia perche' e' necessario che tu impari rapidamente a scrivere i programmi servendoti della tastiera.

4.7 COME SI ESEGUE UN PROGRAMMA

Per eseguire un programma basta scrivere RUN e premere RETURN. L'effetto del comando RUN e' il seguente:
. 1: vengono azzerate tutte le variabili numeriche,
. 2: vengono ridotte a stringhe nulle, di 0 caratteri, tutte le variabili stringa,
. 3: il programma va in esecuzione a partire dalla sua prima istruzione.

Il comando RUN puo' essere scritto anche in altro modo: RUN numero-linea. In questo caso i punti 1 e 2 sono identici, mentre il programma va in esecuzione a partire da numero-linea.

Se vuoi evitare l'effetto dei punti 1 e 2 puoi far partire il programma scrivendo:
GOTO num-linea e premendo RETURN.

Se durante l'esecuzione del programma vedi comparire un messaggio di errore, devi rivedere il listato, e in particolare la linea segnalata come errata, correggere e riprovare. Ricordati che modificando il programma, cioe' entrando in EDITOR, si cancellano i contenuti delle variabili. Vedi il Paragrafo 11.4.

La prova di un programma e' una parte molto importante del lavoro di programmazione. Essa puo' risultare molto gravosa, se non e' stata

svolta bene l'analisi del problema. Inoltre, se un programma e' complesso, devi dedicare molta cura alla preparazione dei casi prova; infatti essi devono essere tali da consentire di provare il programma in tutte le sue parti e nelle condizioni limite.
Per esempio, se nel programma ES5, avessimo omesso l'uso della funzione ABS e eseguito la prova solo con numeri tali che il risultato del calcolo fosse stato positivo, il programma avrebbe in seguito dato errore, la prima volta che i numeri scelti avessero fornito un risultato negativo.

4.8 RIEPILOGO

In questo capitolo sono stati trattati i seguenti argomenti:

.PROGRAMMA PER CALCOLATORE
.LINGUAGGIO BASIC
.LINEE DI PROGRAMMA BASIC
.SCHEMA UTILIZZO MEMORIA
.STRUTTURA ISTRUZIONI BASIC
.ANALISI PROBLEMI
.SCHEMATIZZAZIONE PROBLEMI
.ORGANIZZAZIONE PROGRAMMA
.TIPI DI ISTRUZIONI DEL BASIC
.SCRITTURA DI UN PROGRAMMA
.ISTRUZIONI INPUT, PRINT, LET, GOTO
.FORMATI DI STAMPA E CARATTERI SEPARATORI
.LISTA ED ESECUZIONE DI UN PROGRAMMA

se non ti sembra di avere le idee chiare al riguardo torna indietro.

CAPITOLO 5

GESTIONE DEL PROGRAMMA

5.1 COME SI SCRIVE PIU' FACILMENTE UN PROGRAMMA

Il BASIC 3.5 mette a disposizione alcuni comandi che facilitano la stesura dei programmi. Essi sono: AUTO e RENUMBER.

Il comando AUTO, che ha senso usare solo in modo immediato, consente di fissare l'incremento per i numeri di linea che si scrivono dopo la sua esecuzione. Scrivendo:

AUTO 10

fissi a 10 l'incremento tra i numeri di linea. Dopo l'esecuzione del comando, puoi scrivere una linea di programma iniziando con il numero che desideri; quando premi RETURN, la tua linea viene memorizzata e compare automaticamente sul video il numero della linea successiva (vecchio numero + 10, in questo caso), con il cursore posizionato dopo di esso, e tu puoi cominciare a scrivere la prima istruzione della nuova linea. In sostanza risparmi la fatica di scrivere il numero di linea e eviti di sbagliarlo. Conviene dare come incremento un numero maggiore di 1, in modo di poter inserire nuove linee in caso di bisogno.
Per uscire dalla predisposizione AUTO, basta scrivere AUTO e premere RETURN. Per interrompere la numerazione e passare ad altro basta premere solo RETURN quando compare il numero

della nuova linea; questo pero' non annulla la predisposizione AUTO che rimane attiva e si manifesta quando si ricomincia a scrivere nuovamente una linea di programma o se ne corregge una gia' esistente.

Non devi preoccuparti se dopo aver immesso un programma esso si presenta con numerazione irregolare, cioe' non sempre lo stesso intervallo tra le linee; il programma infatti funziona ugualmente, se non contiene istruzioni errate o errori di impostazione. Quando sei sicuro che il programma funziona, oppure quando lo desideri, puoi rimettere ordine nella numerazione delle linee di programma con il comando RENUMBER.

Il comando RENUMBER, che ha senso usare solo in modo immediato, si puo' scrivere senza far seguire la parola chiave da parametri; in questo caso il programma presente in memoria, viene rinumerato partendo dal numero 10 per la prima linea e incrementando di 10 i numeri di linea. Nella rinumerazione vengono risistemati tutti i richiami a linee preesistenti. La parola chiave RENUMBER puo' essere seguita da 3 parametri:
RENUMBER numeron,inc,numerov
dove:
. numeron, e' il nuovo numero da cui partire,
. inc, e' l'incremento da usare tra i numeri di linea,
. numerov, e' il vecchio numero di linea da cui partire nella rinumerazione.
Se trascuri il primo o il secondo parametro, devi mettere al loro posto una virgola, e vengono presi i valori di default, cioe' 10. La presenza del terzo parametro consente di rinumerare il programma in modo parziale, cioe' partendo da un punto determinato e non modificando quello che viene prima. Non e' consentito usare per il primo parametro un valore uguale a un numero dilinea gia' esistente, e che non viene modificato

per effetto del punto di partenza (terzo parametro).

Durante la stesura di un programma RENUMBER puo' essere usato anche piu' volte. Per esempio a un certo punto vuoi aggiungere tra due istruzioni preesistenti piu' istruzioni di quanto possibile in base alla numerazione; allora usi RENUMBER dando un incremento grande, aggiungi tutto quello che credi e poi usi ancora RENUMBER per fare ordine.

Quando un programma e' terminato e si pensa di non doverlo piu' modificare, e' consigliabile rinumerarlo partendo da 1 e usando un incremento di 1; evitando i numeri grandi si risparmia memoria.
Nel Paragrafo 4.5 abbiamo gia' parlato dei comandi NEW e LIST. Ti suggeriamo ora alcuni accorgimenti che ti aiutano nella stesura di un programma. Se il tuo programma contiene istruzioni simili o uguali, puoi richiamare sul video con LIST n, la linea di modello, poi portarti su di essa e modificarla dove necessario, anche solo nel numero di linea. Sei facilitato dall'uso dei tasti di spostamento del cursore e dalle possibilita' che ti offre l'EDITOR (vedi Paragrafo 1.7).

5.2 MEMORIZZAZIONE E CARICAMENTO DEI PROGRAMMI

E' importante poter conservare i programmi su cassetta per poterli riutilizzare quando servono. L'unita' DATASSETTE 1531 consente di posizionare il nastro al numero di giri desiderato e questo e' utile per un buon utilizzo dello stesso. Ti raccomandiamo nuovamente di mantenere i tasti del registratore in posizione di riposo e di azionarli solo quando richiesto.
Per memorizzare su nastro un programma devi scrivere:
SAVE "nome"

dove "nome" e' il nome che vuoi assegnare al programma per riconoscerlo. Il nome del programma puo' anche essere dato sotto forma di variabile stringa.
Dopo aver premuto RETURN, sul video compare il messaggio:
PRESS PLAY & RECORD ON TAPE
devi eseguire; basta premere RECORD, dato che si abbassa insieme anche PLAY.
Il calcolatore risponde con:
OK
SAVING "nome"
e il video si sbianca diventando del colore del bordo, il nastro gira, si accende l'indicatore luminoso del registratore, dopo poco ricompaiono le scritte precedenti sul video, con in piu':
READY.
si spegne l'indicatore del registratore e il nastro si ferma. A questo punto il tuo programma e' stato registrato ed e' consigliabile rimettere i tasti PLAY e RECORD in posizione di riposo.

Per essere sicuro che la registrazione e' stata buona, devi procedere alla verifica.
Per verificare la registrazione di un programma devi procedere cosi':
. riavvolgere il nastro per posizionarlo all'inizio della registrazione,
. scrivere:
 VERIFY "nome" e premere RETURN,
il calcolatore chiede:
 PRESS PLAY ON TAPE
e dopo la pressione del tasto compare:
 OK
 SEARCHING FOR "nome"
si sbianca il video , dopo un po' compare:
 FOUND "nome"
 VERIFYING

si sbianca nuovamente il video e alla fine compare:
OK, se e' andato tutto bene,
?VERIFY ERROR se la verifica non e' stata buona.
In quest'ultimo caso si deve riavvolgere il nastro al punto giusto e ripetere la procedura di memorizzazione con conseguente verifica.
Puoi omettere il nome dopo VERIFY; in questo caso viene confrontato il primo programma trovato sul nastro con quello presente in memoria. Usando il nome, invece, se vengono incontrati prima altri programmi, essi non vengono confrontati e la verifica avviene solo quando viene trovato il programma con il nome richiesto.

Segue la spiegazione del significato del FLAG che puo' accompagnare le istruzioni SAVE, VERIFY e LOAD. L'argomento e' stato affrontato per completezza, pero', se sei un principiante, ti consigliamo di tralasciarne per il momento la lettura.
Abbiamo considerato la forma piu' semplice del comando SAVE; in realta', dopo il nome del programma, si possono aggiungere altri due parametri, cosi':
SAVE "nome",1,flag
dove 1 e' il numero logico dell'unita' DATASSETTE 1531, e il flag puo' valere 1, 2 o 3, con il seguente significato:
. flag=1, per memorizzare il programma in modo che al momento del LOAD non venga modificato il suo indirizzo di inizio in memoria, ma mantenga quello che era al momento del SAVE,
. flag=2, per memorizzare dopo il programma una segnalazione di "FINE NASTRO",
. flag=3, per ottenere insieme i due precedenti effetti.
Se il flag viene omesso esso vale 0 e ha il significato di memorizzare il programma in modo rilocabile. La RILOCAZIONE di un programma ha questo significato:
. di norma il programma BASIC inizia al byte di

indirizzzo 4097 e questo indirizzo si trova nei byte 43 e 44 (puntatore all'inizio del programma),
. se prima di scrivere un programma sposti il puntatore all'inizio del programma modificando il contenuto dei byte 43 e 44, il programma inizia a un indirizzo diverso da 4097,
. se usi il flag=0 nell'istruzione SAVE il programma viene memorizzato in modo da essere rilocabile,
. se usi il flag=1 o il flag=3 nell'istruzione SAVE il programma viene memorizzato mantenendo l'informazione sul suo indirizzo di inizio, qualunque esso sia, e non e' rilocabile,
. al momento del LOAD, istruzione nella quale e' possibile usare per il flag il valore 1, o il valore 0 (assenza di flag), si ha il seguente comportamento:
.. per flag=0 il programma viene caricato a partire dall'indirizzo di inizio programma che si trova nei byte 43 e 44, rilocandolo o meno a seconda del modo come era stato memorizzato
.. per flag=1 il programma viene caricato a partire dall'indirizzo dove si trovava al momento del SAVE, e questo puo' essere in disaccordo con il valore contenuto nei byte 43 e 44.

Se per memorizzare e' stato usato il flag, l'istruzione VERIFY deve essere scritta in modo tale che non ci sia contrasto con la situazione della memoria e quello che sta sul nastro. Se risulta necessario usare il flag, puoi scrivere cosi':
VERIFY "nome",1,flag.
Se sottointendi il numero dell'unita', viene assunto il valore di default che e' 1. Se vuoi scrivere il flag e non mettere 1, devi usare due virgole.

Nel Paragrafo 0.4 abbiamo visto come si caricano da nastro i programmi della cassetta allegata al libro. Non ripetiamo qui il comando LOAD. Vogliamo solo aggiungere che esso si puo'

scrivere anche cosi':
LOAD "nome",1,flag
dove:
. "nome", puo' anche essere una variabile stringa che contenga il nome del programma,
. 1, e' il numero logico dell'unita' DATASSETTE 1531,
. flag puo' valere 0 (o essere assente) o 1, con i significati sopra spiegati.

5.3 RIEPILOGO

Ci siamo occupati di:

.NUMERAZIONE AUTOMATICA: AUTO
.RINUMERAZIONE LINEE: RENUMBER
.MEMORIZZAZIONE DEI PROGRAMMI SU NASTRO: SAVE
.VERIFICA REGISTRAZIONE: VERIFY
.CARICAMENTO PROGRAMMI DA NASTRO: LOAD

se non hai le idee chiare rivedi questo capitolo dopo aver fatto nuovamente girare CAP5.

CAPITOLO 6

STRUTTURE DEL PROGRAMMA

6.1 STRUTTURA CONDIZIONALE

Le condizioni che il COMMODORE 16 puo' valutare, sono relazioni del tipo:

```
1>2
2<8
A/3=SIN(Z)
```

Ad ogni espressione di questo tipo il COMMODORE 16 sa assegnare il valore VERO o FALSO, esprimendo VERO con il numero -1 e FALSO con il numero 0. Prova ad esempio a scrivere PRINT 1>2 e premi RETURN: la risposta sara' 0, cioe' FALSO; ma se scrivi PRINT 2<8 e premi RETURN la risposta sara' -1 cioe' VERO. Puoi anche assegnare una risposta di questo tipo ad una variabile; ad esempio l'istruzione A=B=C vuole dire: poni A=-1 se B=C, altrimenti poni A=0. Una variabile usata come la variabile A prende il nome di VARIABILE BOOLEANA o LOGICA.
Quando parleremo di CONDIZIONI ci riferiremo quindi a relazioni del tipo A>2 o a variabili booleane.
Gli operatori relazionali del COMMODORE 16 sono 6:

```
=  UGUALE
>  MAGGIORE
<  MINORE
<> DIVERSO
>= MAGGIORE O UGUALE
<= MINORE O UGUALE
```

Queste relazioni valgono anche per stringhe e costanti alfanumeriche:

"A"<"B" VERO (la lettera A viene prima della B)
"WA"<"RZ" FALSO (la lettera W viena dopo la R)

Piu' condizioni possono essere combinate per formare un'unica condizione con gli operatori logici AND OR e NOT:
COND1 AND COND2: vero se sono vere entrambe
COND1 OR COND2: vero se sono vere o COND1 o COND2 o entrambe
NOT COND1: vero se COND1 e' falsa

ESEMPI:
A>0 AND A<4: VERO se 0<A<4
A<0 OR A>4: VERO se A non appartiene all'intervallo 0-4

NOT A: VERO se la variabile booleana A e' falsa

Se per combinare piu' condizioni usi piu' operatori logici, ricorda che il COMMODORE 16 esegue prima NOT poi AND e quindi OR. Se vuoi farle eseguire con un ordine diverso puoi usare le parentesi:

COND1 AND (COND2 OR COND3)

In questo caso eseguira' prima la OR e quindi la AND.

IF ... THEN ... ELSE
Se vuoi che il COMMODORE 16 esegua un'azione solo se e' verificata una certa condizione, devi usare l'istruzione IF ... THEN.
La parola chiave IF deve essere seguita da una condizione, seguita dalla parola chiave THEN e da una o piu' istruzioni separate da due punti. Tutte le istruzioni che appaiono dopo il THEN sulla linea di programma vengono eseguite solo se la condizione e' verificata:

Ecco alcuni esempi di istruzione IF:

```
IF A THEN COLOR1,7,3: PRINT "PIPPO"
```

oppure

```
IF X>0 AND X<10 THEN PRINT "0<X<10"
```

Nel primo caso A e' una variabile booleana, nel secondo X e' una variabile reale.
Nel primo caso, se A e' vero (cioe' A=-1)il COMMODORE 16 colora di blu' il cursore e scrive "PIPPO", se invece A e' falso (cioe' A=0), non scrive nulla e non cambia il colore del cursore.
Se A vale un numero diverso da 0 e da -1 il COMMODORE 16 considera A vera ma anche NOT A viene considerata vera.
Questo capita perche' il BASIC non possiede delle vere e proprie variabili booleane (i cui valori siano cioe' solo VERO o FALSO), ma permette che il programmatore usi le variabili numeriche come booleane.
Assegna quindi i valori 0 e -1 (intendendo FALSO e VERO) alle variabili numeriche usate come booleane, e intende FALSE le variabili numeriche che valgono 0 e VERE le variabili numeriche che hanno un valore diverso da 0. L'operazione NOT e' un'operatore logico che puo' essere applicato anche su numeri interi e che per 0 vale -1 ma per 5, ad esempio, vale -6: succede quindi che 5 e NOT 5 siano entrambi VERI (sono entrambi diversi da 0). Nel BASIC del COMMODORE 16, applicando NOT a un numero reale R, come risultato si ottiene NOT INT (R). I numeri reali per i quali NOT R = 0 sono quindi quelli compresi tra 0 (escluso) e -1.
Si puo' completare una frase IF con la parola chiave ELSE: deve essere posta dopo l'istruzione (o le istruzioni) che segue THEN, preceduta da due punti e seguita dalla o dalle istruzioni che si vogliono far eseguire se la condizione non e' verificata: ad esempio

```
500 IF A>2 THEN GOSUB 1000: ELSE GOSUB 2000
510 ...
```

Se A>2 esegue la routine in 1000 e dopo prosegue da 510.
Se A<=2 esegue la routine in 2000 e dopo prosegue da 510.

6.2 I CICLI

Come hai gia' visto nei capitoli precedenti, e' possibile far eseguire al tuo COMMODORE 16 una o piu' azioni infinite volte, come in ES6.1:

```
0 REM ES6.1
10 PRINT"CIAO"
20 GOTO10
```

Ma possiamo voler far ripetere al calcolatore un'azione, o piu' azioni, per un numero prefissato di volte o fino a che una certa condizione venga verificata: per far cio' usereremo le istruzioni BASIC
FOR ... NEXT e
DO ... LOOP

I CICLI FOR ... NEXT
Le istruzioni FOR ... NEXT servono a far eseguire al calcolatore una o piu' azioni per un numero prefissato di volte. Esse debbono essere poste rispettivamente all'inizio e alla fine della serie di istruzioni che si vuole far ripetere.
La FRASE FOR puo' essere di questo tipo:

```
FOR I=1 TO 10
```

La lettera I e' il nome di una variabile di controllo, e puo' essere sostituita da un qualunque altro nome valido per una variabile numerica, i numeri 1 e 10 sono il valore inizia-

le e finale della variabile: questo ciclo viene quindi ripetuto 10 volte e la variabile I assume i valori da 1 a 10.
La frase che serve per chiudere la serie di azioni che vogliamo far eseguire per dieci volte e' NEXT I o, piu' semplicemente NEXT.
All'uscita dal ciclo la variabile di controllo contiene l'ultimo valore raggiunto, quello per il quale il ciclo non e' stato eseguito.
Se, per esempio, vogliamo far scrivere al COMMODORE 16 i numeri da 1 a 10, dobbiamo programmarlo nel seguente modo:

```
0 REM ES6.2
10 FOR I=1 TO 10
20 PRINT I
30 NEXT I
```

Il COMMODORE 16, quando trova il comando FOR capisce che deve iniziare un ciclo la cui variabile di controllo e' I; ad essa assegna il valore iniziale (nel nostro caso 1) e passa quindi ad eseguire la prossima istruzione. Quando incontra NEXT aggiunge alla variabile di controllo 1 e, se I e' minore o uguale al valore finale torna all'istruzione che segue la frase FOR (nel nostro caso PRINT I), altrimenti prosegue. La frase FOR puo' essere completata dalla parola chiave STEP: questa, se usata, deve essere seguita da un numero reale sia positivo che negativo che viene sommato alla variabile di controllo quando il COMMODORE 16 trova l'istruzione NEXT. Questo numero si chiama INCREMENTO. Se l'incremento e' negativo il numero iniziale deve essere maggiore del numero finale e, arrivato all'istruzione NEXT, il calcolatore controlla che I sia ancora MAGGIORE o uguale al valore finale per tornare al ciclo.
Per capire meglio abbiamo preparato il programma ES6.3:

```
0 REM ES6.3
10 FOR I=10 TO 9 STEP -0.6
20 PRINT I
30 NEXT I
40 PRINT "FINE"
```

Cerchiamo di capire passo per passo che operazioni compie il COMMODORE 16 quando esegue ES6.3, commentando le linee:

. 10: inizializza il ciclo e pone la variabile I=10
. 20: scrive il valore di I, cioe' 10
. 30: incrementa la variabile di controllo di -0.6 cioe' I=10+(-0.6)=9.4 e poiche' I e' maggiore di 9 (valore finale) torna alla linea 20
. 20: scrive il valore di I, cioe' 9.4
. 30: incrementa la variabile di controllo di -0.6 cioe' I=9.4+(-0.6)=8.8 e poiche' I e' minore di 9 (valore finale) prosegue
. 40: scrive FINE e termina.

Se il valore iniziale e' maggiore di quello finale e l'incremento e' positivo (o se il valore iniziale e' minore di quello finale e l'incremento e' negativo), le istruzioni comprese tra FOR e NEXT vengono eseguite una volta.

Si possono far compiere piu' cicli FOR ... NEXT posti uno dentro l'altro come in ES6.4

```
0 REM ES6.4
10 FOR I=1 TO 10
20 FOR J=1 TO 30
30 NEXT J
40 NEXT I
```

Si possono "inscatolare" al massimo 10 cicli uno dentro all'altro. Se si tenta di inserire l'undicesimo, quando incontra l'inizializzazione di questo, il COMMODORE 16 da' il messaggio di

errore OUT OF MEMORY, poiche' ha gia' occupato tutta l'area in cui puo' memorizzare il numero della prima linea del ciclo, il valore finale della variabile di controllo e l'incremento.
E' sbagliato, invece, far eseguire dei cicli concatenati come in ES6.5

```
0 REM ES6.5
10 FOR I=1 TO 10
20 FOR J=1 TO 30
30 NEXT I
40 NEXT J
```

In un caso come questo, se dopo l'istruzione NEXT e' indicata la variabile a cui si riferisce, il COMMODORE 16 da' il messaggio di errore NEXT WITHOUT FOR; altrimenti eseguira' i due cicli come se fossero "inscatolati" nel modo corretto. Ti consigliamo quindi di scrivere sempre il nome della variabile di controllo a cui si riferisce un NEXT: in questo modo se hai inanellato male due cicli FOR ... NEXT il COMMODORE 16 ti segnala l'errore invece di comportarsi in maniera diversa da come tu pensavi. Un altro buon motivo per scrivere la variabile di controllo dopo un NEXT e' la LEGGIBILITA' del programma: se cioe' cerchi di capire cosa volevi far fare al tuo COMMODORE 16 con un programma scritto tre mesi fa', ti puo' essere di grande aiuto aver scritto, dopo ogni NEXT la variabile a cui ti riferivi, anche se al momento ti sembrava inutile.

Per chiudere due cicli 'inscatolati' correttamente puoi usare anche la forma NEXT I,J che e' perfettemente equivalente a NEXT I: NEXT J.

Un'ultima osservazione su questo genere di cicli: il valore iniziale, il valore finale e l'incremento possono essere numeri interi o reali, variabili, o espressioni matematiche del tipo:

```
0 REM ES6.6
10 FOR I=LOG(A+3) TO EXP(8) STEP SIN(7.4)
```

I CICLI DO ... LOOP
Le istruzioni DO ... LOOP permettono di far eseguire al COMMODORE 16 una o piu' azioni, fino a che una condizione non venga verificata.
Se usate come in ES6.7, queste istruzioni provocano la ripetizione infinita delle istruzioni comprese tra DO e LOOP.

```
0 REM ES6.7
10 DO
20 PRINT "CIAO"
30 LOOP
```

Le istruzioni che servono per uscire dal ciclo sono UNTIL, WHILE e EXIT.

UNTIL deve essere usato dopo DO o dopo LOOP e deve essere seguito da una condizione, come esemplificato in ES6.8.a e in ES6.8.b.

```
0 REM ES6.8.A
10 DO UNTIL A$<>""
20 GETA$
30 LOOP
```

```
0 REM ES6.8.B
10 DO
20 GETA$
30 LOOP UNTIL A$<>""
```

Se la condizione A$<>"" non viene verificata, il COMMODORE 16 continua a ciclare come se non ci fosse UNTIL; ma appena la variabile A$ contiene qualche cosa (perche' e' stato premuto un tasto), il COMMODORE 16 esce dal ciclo e si ferma. Con UNTIL, quindi il calcolatore cicla

FINO A CHE la condizione non sia verificata. La differenza tra il programma ES6.8.a e il programma ES6.8.b e' che, mettendo UNTIL dopo LOOP, le istruzioni tra DO e LOOP vengono eseguite almeno una volta (come nei cicli FOR ... NEXT); usando invece UNTIL dopo DO, le istruzioni tra DO e LOOP possono non essere eseguite neanche una volta (se si entra nel ciclo quando la condizione e' gia' verificata).
Anche WHILE deve essere usato dopo DO o dopo LOOP e, come UNTIL, deve essere seguito da una condizione. A differenza di UNTIL il calcolatore cicla MENTRE la condizione che segue WHILE e' vera , cioe' fino a quando la condizione diventa falsa. In altre parole WHILE COND e' come dire UNTIL NOT COND.
L'ultima istruzione che serve per uscire dai cicli DO LOOP e' EXIT. Puo' essere messa come una qualunque istruzione tra DO e LOOP e, quando viene trovata, l'esecuzione del programma salta all'istruzione dopo LOOP, cioe' esce dal ciclo. Conviene quindi usare l'istruzione EXIT cosi':

```
IF COND THEN EXIT
```

Il suo effetto e' cosi quello di UNTIL con la differenza che puo' essere messo in un qualunque punto del ciclo: in questo modo il ciclo puo' essere abbandonato dopo aver eseguito solo una parte delle istruzioni che lo compongono.

Valgono le stesse norme di nidificamento ('inscatolamento') dei cicli FOR ... NEXT.
Puoi usare al massimo 39 cicli DO ... LOOP "inscatolati".
Nell'Appendice A in Figura A.2 sono riportati gli schemi delle istruzioni per il controllo dei cicli.

6.3 I SOTTOPROGRAMMI

Si puo' usare il nome SOTTOPROGRAMMA o SUBROUTINE, con lo stesso significato.
Quando il calcolatore trova l'istruzione GOTO N, va semplicemente a eseguire le istruzioni che trova nella linea N. Se invece trova l'istruzione GOSUB N, prima di saltare alla linea N, memorizza il punto del programma in cui si trova (o memorizza che l'istruzione e' stata data in modo diretto). Quando il COMMODORE 16 trova l'istruzione RETURN torna all'istruzione immediatamente successiva all'istruzione GOSUB. Questo fatto e' molto utile nel caso in cui, in un programma, devi far eseguire molto spesso un gruppo di istruzioni. Ad esempio, se devi sistemare i colori dello schermo in piu' punti di un programma, puoi organizzare il tuo programma come segue.

```
10 ........
20 ........
..
..
100 GOSUB 1000
110 .......
...
...
230 GOSUB 1000
240 .......
...
...
280 GOSUB 1000
...
...
400 END

1000 COLOR 0,1
1010 COLOR 1,7,4
1020 COLOR 4,1
1030 RETURN
```

L'insieme di istruzioni dalla linea 1000 alla

linea 1030 prendono il nome di SUBROUTINE o SOTTOPROGRAMMA. Le SUBROUTINE sono molto utili anche nei casi in cui devi far eseguire molte istruzioni al calcolatore solo se una condizione e' verificata, come in ES6.9.

```
0 REM ES6.9
10 DO
20 INPUT"COME TI CHIAMI ";A$
30 IFA$="FINE"THEN END
40 IFA$="PIPPO"THENGOSUB1000:ELSEGOSUB2000
50 LOOP
1000 COLOR0,7,6
1020 COLOR1,2,7
1030 COLOR4,7,6
1040 PRINT"␣CIAO "A$
1050 PRINT"SONO MOLTO CONTENTO DI VEDERTI"
1060 RETURN
2000 COLOR0,1
2010 COLOR1,5,3
2020 COLOR4,1
2040 PRINT"␣VAI VIA "A$
2050 PRINT"OGGI SONO DI MALUMORE"
2060 RETURN
```

In questo programma le ROUTINE sono 2:
.La prima, dalla linea 1000 alla linea 1060, viene eseguita se la stringa ricevuta nella linea 20 e' "PIPPO".
.La seconda, dalla linea 2000 alla linea 2060, viene eseguita se la stringa ricevuta nella linea 20 non e' "PIPPO".

Come per i cicli, puoi nidificare anche le ROUTINE; e come per i cicli DO ... LOOP, il numero massimo e' di 39 ROUTINE "inscatolate". L'area usata per memorizzare le linee a cui deve tornare l'esecuzione del programma dopo una SUBROUTINE e' la stessa che viene utilizzata per i cicli FOR ... NEXT e per memorizzare la prima linea dei cicli DO ... LOOP: quest'area di memoria si chiama STACK.

Il numero massimo di SUBROUTINE nidificabili e' dunque 39, se il piu' interno di queste non contiene a sua volta uno o piu' cicli FOR ... NEXT o DO ... LOOP. Il decimo ciclo FOR ... NEXT puo' contenere ancora 3 tra GOSUB e DO ... LOOP nidificati.

6.4 I SALTI CALCOLATI

L'istruzione GOTO provoca un salto senza condizioni in un determinato punto di un programma. L'istruzione IF...THEN...ELSE provoca salti sotto condizione. Esistono due istruzioni:
ON...GOTO e ON...GOSUB
che provocano salti in base a un calcolo, cioe' al valore che ha al momento dell'esecuzione un'espressione numerica, che in particolare puo' essere solo una variabile.
Esse si scrivono:

```
ON esp GOTO n1,n2,n3,...,ni
ON esp GOSUB n1,n2,n3,...,ni
```

e agiscono cosi':
se l'espressione vale 1 il salto avviene a n1, primo numero di linea citato dopo GOTO o GOSUB, se essa vale 2 al secondo, se essa vale i all'i-esimo. Se il valore dell'espressione risulta 0 o supera il numero dei numeri di linea presenti il programma prosegue dalla linea seguente. Se il valore risulta negativo si ottiene un messaggio di errore.
Nel caso di GOSUB viene eseguito il sottoprogramma che inizia in ni e al RETURN viene eseguita l'istruzione successiva a ON...GOSUB.

Vediamo un esempio:

```
1 REM ES6.10
10 PRINT"▚    SCELTA PROCEDURA▚▚▚"
15 PRINT"   1: CALCOLO":PRINT"   2: VERIFICA"
20 PRINT"   3: STAMPA":PRINT"   9: FINE"
30 GETKEYA$:IFA$<"1"ORA$>"3"ANDA$<>"9"THEN30
35 ON VAL(A$) GOSUB 100,200,300,900
100 PRINT100:STOP
200 PRINT200:STOP
300 PRINT300:STOP
900 PRINT900:STOP
```

Nel programma ES6.10 viene proposto un menu' di scelta di procedure; viene ricevuta la risposta in A$ con GETKEY, essa viene controllata e accettata solo se corretta. Alla linea 35 viene scelto il sottoprogramma da eseguire in base al valore di A$. Alle linee 100, 200, 300 e 900 viene stampato un numero corrispondente al numero di linea e si ha uno STOP. Sarebbe poco ortodosso terminare un sottoprogramma con STOP, ma l'esempio vuole solo mettere in evidenza l'istruzione ON...GOSUB.

6.5 RIEPILOGO

In questo capitolo abbiamo parlato di:

.CONDIZIONI
.OPERATORI LOGICI
.IF...THEN...: ELSE...
.FOR...NEXT
.STEP
.DO...LOOP (UNTIL, WHILE, EXIT)
.GOSUB
.RETURN
.NIDIFICAMENTO
.ON...GOTO e ON...GOSUB

se non hai le idee chiare al riguardo torna indietro.

CAPITOLO 7

LA GRAFICA

7.1 ALTA RISOLUZIONE E MULTICOLORE.

Al momento dell'accensione, il video puo' mostrare caratteri alfanumerici e grafici disposti in 1000 possibili CASELLE. Ogni casella e' divisa a sua volta in 64 CASELLINE disposte su 8 righe di 8 elementi ciascuna. Le caselline prendono il nome di PIXEL. Se in una casella coloriamo opportunamente alcuni pixel, possiamo ottenere il disegno di una lettera o di un numero o di un simbolo grafico. Ad esempio, possiamo ottenere la lettera "A" in questo modo:

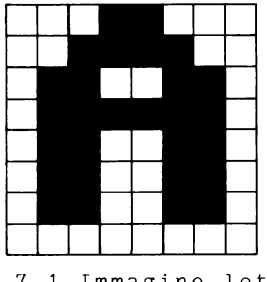


Figura 7.1 Immagine lettera A

I pixel contenuti nel video del COMMODORE 16 sono quindi 64*1000=64000 disposti su 25*8=200 righe di 40*8=320 elementi.

Quando accendi il calcolatore non puoi controllare i pixel individualmente: non puoi decidere se un singolo pixel deve essere del colore dello sfondo o no; puoi solo far visualizzare i simboli (alfanumerici o grafici) dei due set di caratteri del COMMODORE 16 nelle 1000 caselle del video. Chiamiamo questo modo di visualizzazione MODO TESTO. Oltre al modo testo esistono il MODO ALTA RISOLUZIONE e il MODO MULTICOLORE. Nel primo puoi controllare ogni pixel del video, nel secondo controlli solamente 160 pixel per riga (ogni pixel e' largo come due pixel in alta risoluzione), ma puoi gestire meglio il colore (come nell'esempio del programma CAP7 della cassetta).

7.2 IL PENNINO (O CURSORE GRAFICO).

Quando disegni su un foglio di carta, sposti la penna a volte premendola sul foglio, altre tenendola sollevata. Anche il tuo COMMODORE 16 usa un immaginario pennino per disegnare. All'accensione o dopo un'istruzione di pulizia dello schermo, il pennino e' posizionato nell'angolo in alto a sinistra dello schermo (il punto 0,0). Puoi, con l'istruzione LOCATE X,Y, posizionare il pennino nell'X-esimo punto dell'Y-esima riga, tenendolo SOLLEVATO (cioe' senza lasciare la traccia). Tracciando rette, circonferenze, poligoni o altre figure il pennino viene spostato e viene lasciato sull'ultimo punto disegnato. Il pennino serve come punto sottointeso in alcune istruzioni (ad esempio si puo' tracciare una retta dal pennino ad una altro punto o disegnare un rettangolo che abbia come vertice il punto occupato dal pennino); inoltre il pennino puo' essere usato come origine di un sistema di riferimento di coordinate relative (vedi prossimo paragrafo).

7.3 SISTEMI DI COORDINATE

Il modo piu' spontaneo di indicare un punto dello schermo e' quello di dire a quale riga e quale colonna appartiene: il punto in alto a destra e', per esempio, il punto della colonna 319, riga 0 (piu' brevemente punto 319,0). E' proprio con la coppia di numeri 319,0 che ci si riferisce, normalmente, al punto in alto a destra. Questo modo e' sicuramente molto immediato, ma puo' essere utile, a volte, indicare un punto RELATIVAMENTE al pennino. Puo' essere utile ad esempio voler indicare il punto che sta sopra al pennino di 7 punti: per far cio' basta indicare il punto +0,-7. Il COMMODORE 16, infatti, capisce che vuoi indicare un punto relativamente al cursore grafico, se poni un segno + o - prima delle coordinate: un + prima del numero di colonna indica un punto piu' a destra del pennino; un + prima del numero di riga indica un punto piu' in basso del pennino. E possibile anche indicare la colonna di un punto in maniera assoluta e la riga in maniera relativa (o viceversa). Ad esempio il punto 5,-9 e' il punto della colonna 5 che appartiene alla nona riga sopra il cursore grafico. Un' altro modo di indicare un punto relativamente al pennino e quello di dare una distanza e un angolo, cioe' di indicare il punto usando un sistema di COORDINATE POLARI. Per far cio' devi separare le due coordinate con un punto e virgola (anziche' una virgola). L'angolo deve essere misurato in gradi, in senso orario partendo dalla semiretta dal pennino verso l'alto. Ad esempio il punto 30;90 e' trenta punti piu' a destra del pennino.

7.4 MODI GRAFICI (L'ISTRUZIONE GRAPHIC)

Nel modo alta risoluzione e multicolore il calcolatore usa una zona di memoria, diversa da quella che usa in modo testo, per ricordare il

contenuto del video. E' possibile, quindi, passare da un modo di visualizzazione all'altro senza perdere il contenuto delle videate di testo o di grafica. L'istruzione che permette di passare da un modo di visualizzazione all'altro e' GRAPHIC:

GRAPHIC 0 visualizza in modo testo.

GRAPHIC 1 visualizza in modo alta risoluzione.

GRAPHIC 2 visualizza i quattro quinti in alto in modo alta risoluzione e il quinto rimanente in modo testo.

GRAPHIC 3 visualizza in modo multicolore.

GRAPHIC 4 visualizza i quattro quinti in alto in modo multicolore e il quinto rimanente in modo testo.

Aggiungendo ,1 a questi ordini ottieni la pulizia dello schermo (grafico se passi a un modo grafico, testo se passi al modo testo, entrambi se passi a un modo misto). Puoi pulire lo schermo senza cambiare modo grafico con l'istruzione SCNCLR.
Se non usi mai un modo di visualizzazione grafico, il COMMODORE 16 non usa memoria per la pagina grafica; quando pero' entri in modo grafico il COMMODORE deve RUBARE 10230 byte di memoria al BASIC, che rimane con solo 2045 byte. Se torni al modo testo con l'istruzione GRAPHIC 0, i byte rubati non vengono restituiti al BASIC; vengono pero' restituiti tornando al modo testo con l'istruzione GRAPHIC CLR.

7.5 I COLORI NEI MODI GRAFICI

Nel Capitolo 2 abbiamo visto la funzione dell'istruzione COLOR, ma non abbiamo chiarito

quali sono le zone 2 e 3. In modo alta risoluzione e' consentito disegnare nel colore dello sfondo (0) o nel colore dei caratteri (1); in modo multicolore, invece, hai a disposizione altri due colori, che corrispondono alle zone 2 e 3. Chiamiamo i colori 1, 2, 3: inchiostro 1, inchiostro 2 e inchiostro 3.
Se, con l'istruzione COLOR, cambi il colore dell'inchiostro 3, tutti i disegni gia' fatti sul video con l'inchiostro 3, cambiano immediatamente colore. Se, invece, cambi il colore dell'inchiostro 2 (o 1), solo i disegni che d'ora in poi farai con l'inchiostro 2 (o 1), avranno un altro colore.

7.6 LE ISTRUZIONI GRAFICHE

Nei prossimi paragrafi descriveremo una serie di funzioni che agiscono sulla pagina grafica. Tutte queste funzioni saranno seguite da una serie di parametri, il primo dei quali e' il colore (0 = colore di sfondo, 1 = inchiostro 1, 2 = inchiostro 2, 3 = inchiostro 3). Fanno eccezione a questa regola le istruzioni SSHAPE e GSHAPE. Spesso si possono sottointendere alcuni parametri: per fare questo basta non scrivere il parametro, lasciando la punteggiatura richiesta. Quando il parametro sottointeso e' un punto devi sottointendere le coordinate e il separatore (virgola per coordinate cartesiane, punto e virgola per coordinate polari). Se usi istruzioni grafiche mentre sei in modo testo e non e' occupata la memoria per la pagina grafica (cioe' non sei mai entrato in modo grafico da quando hai acceso il calcolatore oppure hai usato l'istruzione GRAPHICCLR), il COMMODORE 16 ti risponde con il messaggio di errore
?NO GRAPHICS AREA ERROR
Se, invece, la memoria per la pagina grafica e' stata occupata, le istruzioni grafiche, anche se date in modo testo, vengono eseguite ugualmente

sulla pagina grafica che viene considerata ad alta risoluzione (solo che tu non ne vedi l'effetto subito): se usi, infatti, il colore 2 o 3 (propri del modo multicolore) il COMMODORE 16 ti risponde con il messaggio di errore
?ILLEGAL QUANTITY ERROR
come quando e' in modo alta risoluzione. Questi messaggi di errore non vengono dati se l'istruzione e' CHAR (vedi Paragrafo 7.11).

7.7 PUNTI E RETTE (L'ISTRUZIONE DRAW)

Consideriamo ora una delle piu' versatili istruzioni grafiche del COMMODORE 16: DRAW. Questa istruzione consente di disegnare o cancellare punti, rette e linee spezzate. Nella forma completa l'istruzione deve essere seguita dal colore, le coordinate di un punto, la parola chiave TO e le coordinate di un secondo punto: l'istruzione traccia una retta del colore indicato dal primo al secondo punto. Ad esempio:
DRAW 2,0,0 TO 100,100
disegna una linea del colore dell'inchiostro 2, da 0,0 a 100,100
DRAW 0,0,0 TO 100,100
cancella la linea appena disegnata (disegnandone una del colore dello sfondo).
Omettendo il primo parametro il COMMODORE 16 assume il colore uguale all'inchiostro 1:
DRAW,12;100 TO +50,-20
disegna una retta di colore 1.
Omettendo il primo punto il COMMODORE 16 traccia una retta dal cursore grafico al punto indicato. Poiche' il pennino si trova, dopo un DRAW, sul secondo punto, le istruzioni
LOCATE0,0:DRAW TO 10,10:DRAW TO 10,20:DRAW TO 20,10
disegnano una linea spezzata di colore 1 i cui vertici sono i punti indicati nelle istruzioni.

Se, infine, ometti il secondo punto il COMMODORE 16 disegna solo il primo punto (ove lascia il pennino).

7.8 CIRCONFERENZE, ELLISSI E POLIGONI (L'ISTRUZIONE CIRCLE)

Il modo piu' semplice per usare CIRCLE e' quello in cui indichi solo il colore, le coordinate del centro, la lunghezza del raggio.
CIRCLE2,100,100,30
disegna una circonferenza di colore 2 centrata in 100,100 di raggio 30.
sottointendendo il colore ottieni il colore dell'inchiostro 1, sottointendendo il centro la circonferenza viene centrata sul pennino; il raggio non puo' essere sottointeso.
Gli altri parametri che possono seguire sono:
RAGGIO Y: e' il semiasse verticale di un'ellissi il cui semiasse orizzontale sia lungo quanto il raggio. Se non c'e' viene considerato uguale al raggio X. In modo multicolore devi ricordare che un pixel in orizzontale ne vale due in verticale.
ANGOLO DI PARTENZA: normalmente il calcolatore comincia a disegnare la circonferenza dal punto a 0 gradi dalla semiretta verso l'alto uscente dal centro e la finisce nel punto a 360 gradi (dove viene lasciato il pennino). Se viene dato questo parametro il COMMODORE 16 comincera' a disegnare la circonferenza da quella posizione.
ANGOLO FINALE: l'angolo raggiunto il quale il calcolatore smette di disegnare la circonferenza. Grazie a questo e al precedente parametro puoi disegnare archi di circonferenza.
ROTAZIONE: l' angolo (in senso orario) di cui viene ruotata la figura:
CIRCLE 1,100,100,60,30,,,45
disegna una éllissi il cui asse maggiore e' ruotato di 45 gradi.

Questa operazione viene eseguita male in modo multicolore: infatti il calcolatore non tiene conto della forma asimmetrica dei pixel in questo modo, per cui la figura risulta allungata.
LATI: volendo disegnare un poligono regolare di N lati devi usare come ultimo parametro 360/N:
CIRCLE,100,100,50,,,,,360/6
disegna un esagono regolare.

7.9 L'ISTRUZIONE PAINT

Se vuoi dipingere un'area, ti basta dare al COMMODORE 16 l'istruzione PAINT seguita dal colore che vuoi usare e dalle coordinate di un punto contenuto nell'area che vuoi dipingere
CIRCLE,100,100,50:PAINT,100,100
disegna un cerchio pieno.
Se sottointendi il punto che individua l'area, il calcolatore parte a dipingere dal pennino.
Dopo PAINT il pennino viene lasciato sul punto che individua l'area (punto iniziale).

Puoi aggiungere un ultimo parametro: se vale 1 il calcolatore considera che l'area e' delimitata da un qualunque colore diverso dallo sfondo. Se e' 0 il colore che delimita l'area e' solo il colore 1.

7.10 DISEGNARE RETTANGOLI (L'ISTRUZIONE BOX)

Con l'istruzione BOX puoi disegnare un rettangolo semplicemente dando il colore e due vertici opposti del rettangolo. Il colore sottointeso e' il colore dell'inchiostro 1, il secondo angolo, se sottointeso e' il pennino.
Dopo questa istruzione il pennino viene lasciato sul secondo angolo.
Possono seguire altri due parametri: il primo indica la rotazione (in senso orario, espressa in gradi). Se il secondo e' 1 il rettangolo viene dipinto.

7.11 SCRIVERE CARATTERI IN MODO GRAFICO (L'ISTRUZIONE CHAR)

L'istruzione CHAR e' un'istruzione che vale sia in modo grafico che in modo testo. Scrive su un video pittosto che l'altro a seconda del modo grafico in cui si trova il calcolatore. Nei modi MISTI agisce solo sulla pagina grafica.
Con questa istruzione puoi scrivere tutte le parole che vuoi nella posizione che preferisci.
CHAR1,10,10,"PIPPO"
scrive PIPPO in colore 1 partendo dalla colonna 10, riga 10 in modo testo, colonna 80, riga 80 in modo alta risoluzione. In modo multicolore i risultati sono poco apprezzabili.
Puoi aggiungere a CHAR un ulteriore parametro: se e' 1 la scritta viene stampata in campo inverso.
L'istruzione CHAR accetta anche una serie di CHR$ (separati da +) come stringa. In modo testo vengono stampati normalmente, in modo grafico i caratteri di controllo vengono scritti come se fossero tra virgolette, ma in campo diretto. Il CHR$(255) non e' π ma un simbolo grafico.

7.12 TRASFERIRE PARTI DI SCHERMO GRAFICO (LE ISTRUZIONI SSHAPE E GSHAPE)

Il COMMODORE 16 ti da' la possibilita' di memorizzare una parte rettangolare della pagina grafica in una variabile stringa. Per fare cio' devi usare l'istruzione SSHAPE: tale istruzione deve essere seguita dal nome della variabile in cui si vuole salvare la zona e dai vertici che delimitano il rettangolo (come in BOX).
Per porre questa sezione in un altro punto della pagina grafica devi usare l'istruzione GSHAPE seguita dal nome della variabile che contiene la zona salvata con SSHAPE, e le coordinate in cui

vuoi porre l'angolo in alto a sinistra del rettangolo.
La grandezza dell' area che puoi memorizzare e' limitata dalla lunghezza massima delle stringhe: 255 caratteri. Se vuoi calcolare la lunghezza della stringa che conterra' l'area devi usare le formule:

in alta risoluzione:
```
 INT((ABS(X1-X2)+1)/8+.99)*(ABS(Y1-Y2)+1)+4
```

in multicolore:
```
 INT((ABS(X1-X2)+1)/4+.99)*(ABS(Y1-Y2)+1)+4
```

Dove X1,Y1,X2,Y2 sono rispettivamente la prima e la seconda coordinata del primo angolo e la prima e la seconda coordinata del secondo angolo.

GSHAPE puo' essere seguita da un ultimo parametro: il modo di porre sul video la zona salvata:
0 pone la figura come e',
1 la pone in campo inverso,
2 la pone facendo la OR con l'area,
3 la pone facendo la AND con l'area,
4 la pone facendo la XOR con l'area.

I modi 2, 3 e 4 possono risultare poco chiari.

Vediamo gli effetti su una pagina ALTA RISOLUZIONE:
modo 2, rimane lo sfondo sulla parte di colore 0 della figura,
modo 3, disegna solo le zone in cui sia lo sfondo che la figura sono di colore 1,
modo 4, disegna in colore 1 su sfondo di colore 0 e in colore 0 su sfondo di colore 1.

E gli effetti su pagina MULTICOLORE:
modo 1 si scambiano colore le parti di colore 0 e 3 e quelle di colore 1 e 2,
mod1 2, 3 e 4:

CS	CF	M2	M3	M4
0	0	0	0	0
0	1	1	0	1
0	2	2	0	2
0	3	3	0	3
1	0	1	0	1
1	1	1	1	0
1	2	3	0	3
1	3	3	1	2
2	0	2	0	2
2	1	3	0	3
2	2	2	2	0
2	3	3	2	1
3	0	3	0	3
3	1	3	1	2
3	2	3	2	1
3	3	3	3	0

(se CS e' il colore dello sfondo in un punto e CF e' il colore della figura nello stesso punto, allora M2 e' il colore che viene visualizzato in quel punto ponendo l'immagine sullo sfondo col modo 2, M3 col modo 3 e M4 col modo 4).

7.13 FUNZIONI GRAFICHE

Il COMMODORE 16 possiede 4 funzioni riguardanti grafica e colore: RGR, RCLR, RLUM, RDOT.

RGR, il cui argomento puo' essere qualunque, torna il modo grafico in cui e' il calcolatore (ricordiamo che il modo grafico si cambia con l'istruzione GRAPHIC).

RCLR torna il colore della zona che viene passata come parametro alla funzione:
RCLR (0) torna il colore dello sfondo,
RCLR (1) torna il colore dell'inchiostro 1,
RCLR (2) torna il colore dell'inchiostro 2,
RCLR (3) torna il colore dell'inchiostro 3,
RCLR (4) torna il colore del bordo.

RLUM torna la luminosita' della zona che viene passata come parametro alla funzione:
RLUM (0) torna la luminosita' dello sfondo,
RLUM (1) torna la luminosita' dell'inchiostro 1,
RLUM (2) torna la luminosita' dell'inchiostro 2,
RLUM (3) torna la luminosita' dell'inchiostro 3,
RLUM (4) torna la luminosita' del bordo.

RDOT fornisce informazioni sul pennino:
RDOT (0) torna la colonna su cui e' posizionato il pennino,
RDOT (1) torna la riga su cui e' posizionato il pennino,
RDOT (2) torna il colore del punto su cui si trova il pennino.

Ecco il listato di un programma che usa la funzione RDOT. Troverai il programma registrato sulla cassetta.

```
0 REM ES7.1
5 COLOR0,1:COLOR4,1:YY=1:NC=40:C=8:TRAP500
10 GRAPHIC1,1
20 GETKEYA$
30 IFA$=CHR$(13)THENXX=0:YY=YY+1:GOTO20
40 IFA$=CHR$(20)THENGOSUB300:GOTO20
50 IFA$=CHR$(27)THENGOSUB400:GOTO20
60 GOSUB200
70 XX=XX+C:IFXX>319-CTHENXX=0:YY=YY+1
80 GOTO20
200 COLOR1,1:CHAR,0,0,A$:COLOR1,7,7
210 FORX=0TOC-1
220 FORY=0TO7
230 LOCATEX*8/C,Y
240 DRAWRDOT(2),XX+X,YY*8+Y
250 NEXTY,X
260 RETURN
300 XX=XX-C:YY=YY+(XX<0):XX=XX-320*(XX<0)
310 A$=" ":GOSUB200:RETURN
400 GETKEYA$,B$:NC=VAL(A$)*10+VAL(B$)
410 C=320/NC:RETURN
500 GRAPHIC0:SCNCLR
```

Lo scopo di questo programma e' quello di scrivere sulla pagina grafica caratteri di diversa larghezza con il seguente metodo: viene stampato, nell'angolo superiore sinistro, in nero, il carattere che si vuole allargare o rimpicciolire; la funzione RDOT sa comunque riconoscere se i punti che formano il carattere sono di inchiostro 1 o di sfondo; una routine riproduce quindi, punto per punto, in blu, nell'attuale posizione del cursore la lettera compressa o allargata. Per decidere quanti caratteri si vogliono visualizzare su ogni riga basta premere il tasto ESC seguito dal numero di caratteri per riga (due cifre). Il programma riconosce ed esegue DEL e RETURN.
Vediamo ora, linea per linea, come funziona il programma:

Linea 0: schermo e bordo neri; riga sulla quale verra' stampato il prossimo carattere (YY) = 1; numero di caratteri per linea (NC) = 40; numero di pixel per ogni carattere (C) = 320/40 = 8. Per TRAP vedi Paragrafo 11.5. In questo caso manda l'esecuzione del programma alla linea 500 quando premi RUN/STOP.
Linea 10: entra in alta risoluzione e pulisce lo schermo.
Linea 20: accetta un carattere da tastiera e lo pone nella variabile A$.
Linea 30: se il tasto premuto e' RETURN (=CHR$(13)) va a capo: cioe' incrementa la riga e pone = 0 la colonna nella quale verra' stampato il prossimo carattere (XX indica il primo pixel da cui dovra' partire la stampa del prossimo carattere). Torna quindi alla linea 20 per chiedere il prossimo carattere.
Linea 40: se il tasto premuto e' DEL (=CHR$(20)) esegue la subroutine in 300 che cancella il carattere precedente. Torna quindi alla linea 20.
Linea 50: se il tasto premuto e' ESC (=CHR$(27)) esegue la subroutine in 400 che accetta un nuo-

vo numero di colonne. Torna quindi alla linea 20.
Linea 60: esegue la routine in 200 che stampa un carattere sullo schermo alta risoluzione.
Linea 70: incrementa di C pixel il prossimo punto di stampa. Controlla che il prossimo carattere non ecceda la linea (XX>319+C): se cio' avviene va a capo.
Linea 80: torna alla linea 20 per ricevere il prossimo carattere.

SUBROUTINE DI STAMPA

Linea 200: scrive in alto a sinistra, in nero, il carattere che si vuole stampare e rimette blu il colore del cursore.
Linea 210: inizializza il ciclo dei punti orizzontali del carattere (se il carattere e' largo 5 pixel il ciclo verra' ripetuto 5 volte).
Linea 220: inizializza il ciclo dei punti verticali.
Linea 230: pone il pennino sul punto del carattere che si vuole valutare.
Linea 240: disegna un punto del colore che si legge sul carattere originale nel posto del carattere modificato.
Linea 250: chiude i due cicli.
Linea 260: fine della subroutine.

SUBROUTINE PER DEL

Linea 300: decrementa l'indicatore di colonna di tanti pixel quanto e' la larghezza attuale di un carattere; se cosi' facendo il numero diventa minore di 0 viene decrementato il numero di linea e aggiunto 320 al numero di colonna.
Linea 310: servendosi della subroutine in 200 stampa uno spazio che cancella il carattere indesiderato.

SUBROUTINE PER ESC

Linea 400: accetta due caratteri da tastiera e li considera un numero di due cifre: pone il

numero di colonne (NC) pari al valore calcolato.
Linea 410: pone la larghezza in pixel dei caratteri (C) = 320/NC.

SUBROUTINE PER STOP

Linea 500: torna in modo testo, pulisce lo schermo e il programma si arresta.

7.14 RIEPILOGO

Gli argomenti trattati in questo capitolo sono:

.MODO TESTO, ALTA RIOLUZIONE E MULTICOLORE
.L'ISTRUZIONE GRAPHIC
.IL PENNINO O CURSORE GRAFICO
.L'ISTRUZIONE LOCATE
.COORDINATE ASSOLUTE RELATIVE E POLARI
.L'ISTRUZIONE GRAPHIC
.L'ISTRUZIONE COLOR
.L'ISTRUZIONE DRAW
.L'ISTRUZIONE CIRCLE
.L'ISTRUZIONE PAINT
.L'ISTRUZIONE BOX
.L'ISTRUZIONE CHAR
.LE ISTRUZIONI SSHAPE E GSHAPE
.LE FUNZIONI RGR, RCLR, RLUM E RDOT.

CAPITOLO 8

L'ANIMAZIONE

8.1 IL MOVIMENTO

Come per i cartoni animati, la televisione e il cinema, il movimento che vedi nei videogiochi (o comunque il movimento generato da un calcolatore) non e' vero movimento, ma una sequenza di figure statiche ognuna delle quali differisce di poco dalla precedente. Un grossolano esempio su come imitare un movimento col calcolatore e' dato dal programma ES8.1:

```
0 REM ES8.1
10 PRINTCHR$(147)
20 FORI=0TO38:CHAR,I,0," ●"
30 FORJ=1TO30
40 NEXTJ,I
50 RUN
```

Questo programma disegna, in uno schermo vuoto, una pallina, ogni volta una posizione piu' a destra della precedente. In questo modo sembra che la pallina corra verso destra. Ecco la spiegazione linea per linea del programma:

Linea 10: pulisce lo schermo.
Linee 20-40: per 39 volte stampano, ogni volta un carattere piu' a destra, uno spazio e una pallina. Lo spazio ha la funzione di cancellare la vecchia pallina. Il ciclo a vuoto di J serve a rallentare l'esecuzione. In questo modo

visualizziamo, abbastanza rapidamente, una dopo l'altra delle figure che differiscono di poco dalle precedenti.
Linea 50: ricomincia.

Il motivo per cui il risultato di questo programma non e' molto soddisfacente e' che la differenza tra un'immagine e l'altra e' troppo grande in rapporto alla grandezza dell'oggetto che si muove.

8.2 UN ESEMPIO DI MOVIMENTO

Proviamo ora a costruire un movimento che ci soddisfi di piu':

```
0 REM ES8.2
10 DIMA$(2):PRINTCHR$(147)
20 A$(0)="/●\":A$(1)="—●—":A$(2)="\●/"
30 FORI=0TO2:CHAR,0,0,A$(I)
40 FORJ=0TO90:NEXTJ,I:GOTO30
```

Con questo esempio imitiamo il battere delle ali di un uccellino. Il risultato, questa volta, e' un po' piu' bello perche' il MOVIMENTO delle ali e' di qualche pixel, mentre prima la pallina si muoveva di un carattere alla volta. Vediamo come funziona il programma:

Linee 10-20: memorizzano le immagini necessarie nel vettore di stringhe A$ e puliscono lo schermo.
Linee 30-40: stampano una dopo l'altra le tre immagini nello stesso posto. Il ciclo a vuoto di J rallenta, il programma prosegue fino a quando non premi STOP.

Per completare questo programma possiamo far VOLARE l'uccellino:

```
0 REM ES8.3
10 DIMA$(2):PRINTCHR$(147)
20 A$(0)="/●\":A$(1)="—●—":A$(2)="\●/"
30 Y=25:FORX=0TO24:Y=Y-1
40 GOSUB100
50 NEXTX
60 RUN
100 PRINTCHR$(147)
110 FORI=0TO2:CHAR,X,Y,A$(I)
120 FORJ=1TO50
130 NEXTJ,I
140 RETURN
```

Combinando il movimento delle ali a quello dell'uccellino si ottiene un risultato abbastanza buono. Il programma funziona cosi':

Linee 10-20: come prima.
Linea 30: posizione verticale dell'uccellino (Y) = 25 (piu' in basso possibile). Inizializza un ciclo da 0 a 24, la cui variabile indice e' X (posizione orizzontale), che decrementa la posizione verticale.
Linea 40: fa volare l'uccellino da una casella alla casella immediatamente sopra a destra.
Linea 50: chiude il ciclo.
Linea 60: lancia nuovamente il programma.

ROUTINE DI MOVIMENTO

Linee 100-140: svolgono la funzione del programma precedente con la differenza che la posizione in cui l'uccellino sbatte le ali e' data dai valori di X e Y.

8.3 RIEPILOGO

In questo capitolo abbiamo trattato l'argomento dell'ANIMAZIONE.

CAPITOLO 9

IL SUONO

9.1 IL SUONO

Il COMMODORE 16 possiede due comandi diretti alla gestione del suono. Le possibilta' sonore del COMMODORE 16 sono abbastanza limitate, poiche' vi sono solamente due generatori sonori, e questi generatori producono solo suoni a onda quadra o a onda casuale (rumore bianco); inoltre non vi sono filtri ne' generatori di inviluppo. L'assenza di tutte queste potenzialita' e' forse un handicap, ma e' senza dubbio vantaggioso per quello che riguarda la facilita' d'uso.
I due comandi che gestiscono il suono sono:
VOL e SOUND.

Con essi e' assai semplice generare suoni ed effetti sonori.
VOL deve essere seguito da un numero compreso tra 0 e 8:
VOL 0 = volume a zero
VOL 8 = volume al massimo

Per generare un suono occorre anche specificarne la forma d'onda, la frequenza e la durata. Tutti questi parametri devono essere forniti all'istruzione SOUND.
SOUND V,F,D
produce una nota musicale col generatore V, la frequenza dipendente da F, e della durata di D sessantesimi di secondo.
-V e' un numero che puo' valere 1, 2 o 3. Se vale 1 il suono viene generato dal generatore 1,

sempre con onda quadra, se vale 2 viene generato dal generatore 2 con onda quadra e se V=3 il suono viene sempre generato dal generatore 2 ma con onda casuale (rumore bianco). Ovviamente le voci 2 e 3 non possono suonare insieme, mentre possono suonare assieme voce 1 e voce 2 o voce 1 e voce 3.

-F e' un numero da cui dipende la frequenza, a cui e' legato dalla seguente espressione:
F=1024-(111840.45/Hertz)
Nell'Appendice D trovi i valori del parametro F per tutte le note musicali. Se sei esigente puoi notare una certa stonatura nei suoni, dovuta al fatto che F e' un numero intero, che quindi e' generalmente approssimato.

-D e' un numero compreso tra 0 e 65535 che esprime la durata della nota in sessantesimi di secondo.

Segue il programma ES9.1 che suona le note contenute in linee DATA; esso e' registrato sulla cassetta.

```
0 REM ES9.1
5 DIMMT(75)
10 RD=2↑(1/12)
20 F=7040
30 FORI=73TO1STEP-1
40 MT(I)=INT(1024.5-111840.45/F)
60 F=F/RD
70 NEXT
85 VOL8
90 DO
100 READA:IFA=0THENEXIT
110 SOUND1,MT(A),5:FORI=1TO100:NEXT
120 LOOP
9000 DATA17,15,17,13,17,12,17,10,17,8,17
9002 DATA6,17,15,13,15,15,13,15,12
9005 DATA15,10,15,8,15,6,15,5,15,13,12,13
9006 DATA13,12,13,10,13,8,13,6,13,5,13,4,13
9007 DATA12,10,12,12
9010 DATA10,12,9,12,7,12,5,12,3,12,1,12
9020 DATA10,8,10,0
```

COMMENTO A ES9.1
Linee 0-70: viene definito il vettore MT che contiene tutti i valori di F in base al numero di nota da suonare. Partendo da una frequenza di 7040 Hertz calcoliamo il valore da porre nel vettore, e una per volta ricaviamo il valore della nota piu' bassa dividendo per RD che assume il valore della radice dodicesima di due. In questo programma la variabile F esprime la frequenza in Hertz della nota di cui vogliamo calcolare l'argomento per SOUND.
Linee 90-120: ciclo in cui vengono letti i dati dalle linee DATA, e sono usati come indici del vettore MT. Il numero esprime la nota musicale secondo la seguente tabella:

DO	40	28	16	4
DO#	41	29	17	5
RE	42	30	18	6
RE#	43	31	19	7
MI	44	32	20	8
FA	45	33	21	9
FA#	46	34	22	10
SOL	47	35	23	11
SOL#	48	36	24	12
LA	49	37	25	13
LA#	50	38	26	14
SI	51	39	27	15
OTTAVA	4	3	2	1

Il vettore MT ha 76 elementi. Ti consigliamo di non usare gli elementi piu' alti, perche' rappresentano frequenze un po' troppo alte per il generatore di suoni.

9.2 RIEPILOGO

Abbiamo trattato le istruzioni VOL e SOUND.

CAPITOLO 10

ALTRE ISTRUZIONI BASIC

10.1 LE VARIABILI CON INDICE

Fino ad ora ci siamo occupati di variabili singole (Capitoli 3 e 4); nella programmazione risulta spesso utile poter riferire con lo stesso nome un GRUPPO DI VARIABILI. Sono disponibili per questo le VARIABILI CON INDICE, chiamate anche MATRICI o VETTORI (ARRAY in inglese). Si tratta di una struttura di dati nella quale si usa lo stesso nome per rappresentare un gruppo di variabili; per identificare un elemento del gruppo si usano uno o piu' numeri, detti indici, che forniscono la posizione di quell'elemento nel gruppo.
Vediamo un esempio: consideriamo un vettore di variabili di tipo numerico, di nome A. Il primo elemento di tale gruppo si chiama A(0), e ha le stesse proprieta' di ogni altra variabile di tipo numerico. Il secondo elemento si chiama A(1), e il suo valore e' completamente indipendente dal valore di A(0) e di tutti gli altri elementi del vettore. A(0) e A(1) sono due variabili diverse. Prova a eseguire le seguenti linee:

```
A(0)=10:A(1)=3.14
PRINT A(0);A(1)
```

Il calcolatore stampa 10 e 3.14. Un'osservazione, a questo punto, potrebbe essere questa: "Che cos'hanno di diverso i vettori dalle

variabili normali? Anche la variabile A0 e' completamente separata da A1, e si risparmia di scrivere le parentesi!". La differenza c'e', ed e' importante. L'indice contenuto tra parentesi infatti puo' essere un numero o una variabile numerica, di cui viene considerata solo la parte intera:

K=1:PRINT A(K)
stampa ancora il valore di A(1), mentre:
K=1:PRINT AK
non stampa il valore della variabile A1, ma quello della variabile AK.
Esistono sul COMMODORE 16 tre tipi diversi di variabili con indice:
-MATRICI di variabili di tipo REALE (A)
-MATRICI di variabili di tipo INTERO (A%)
-MATRICI di variabili di tipo STRINGA (A$)
Una matrice puo' avere piu' di un indice, ad esepio B(3,2), che e' diverso da B(2,3), ha due indici. Non esistono limiti sul numero di indici che puo' avere una matrice, tranne il fatto che una matrice con molti indici occupa molta memoria.

Nel seguito useremo il nome MATRICE, che e' il piu' generale; di norma il nome VETTORE viene usato per matrici con un solo indice. A volte usiamo anche il termine inglese ARRAY.

Nel BASIC 3.5 del COMMODORE 16 gli indici iniziano dal valore 0, cioe' il primo elemento del gruppo corrisponde al valore 0 degli indici. L'uso di piu' indici consente di strutturare meglio il gruppo di variabili; per esempio una tabella di dati viene chiaramente definita con variabili a due indici, l'indice di riga, il primo, e l'indice di colonna, il secondo.

Quando si usano le matrici, bisogna definire alcuni parametri:
. il nome della matrice,
. il tipo di variabile,
. il numero di indici che individuano un elemento,
. il massimo valore che puo' raggiungere ogni indice, cioe' l'estensione del gruppo.
Tutte queste informazioni devono essere fornite al calcolatore prima di usare gli elementi della matrice, attraverso l'istruzione DIM, che e' l'istruzione di definizione delle matrici.
Quando noi eseguiamo:
DIM A(4,7,9)
informiamo il calcolatore che vogliamo definire una matrice di nome A, e che ha come elementi numeri REALI (vedi Capitoli 3 e 4). Inoltre specifichiamo che la matrice ha tre indici (inseriamo infatti tre numeri nelle parentesi, separati da virgole). Infine specifichiamo che useremo:
. per il primo indice i valori da 0 a 4,
. per il secondo valori da 0 a 7,
. per il terzo valori da 0 a 9.
Se nel programma usi per gli indici valori superiori a quelli definiti, o un numero di indici diverso da quello specificato, ottieni il messaggio di errore: BAD SUBSCRIPT.
Per definire la matrice A di interi, con 100 elementi, scriveremo:
DIM A%(99)
Il vantaggio principale nell'usare matrici di tipo INTERO anziche' REALE e' che ogni elemento INTERO occupa 2 BYTE di memoria, mentre un elemento REALE occupa 5 BYTE.
Per definire il vettore QS di STRINGHE, con 65 elementi, scriveremo:
DIM QS$(64)
L'occupazione in memoria di un vettore di questo tipo non e' fissa: ogni elemento occupa 3 BYTE, piu' la lunghezza della stringa. Fai attenzione quando usi grosse matrici di strin-

ghe, perche' potresti avere dei problemi di memoria.
Una volta dimensionata, una matrice non puo' piu' essere cancellata, se non con l'istruzione CLR, che cancella tutte le variabili: al massimo si puo' azzerarne il valore, ma non si puo' liberare completamente la memoria che occupa.
Non e' neanche possibile cambiare le dimensioni di un vettore senza cancellare tutte le variabili (con CLR), altrimenti viene emesso il messaggio REDIM'D ARRAY (vettore ridimensionato).
Se si usa un vettore prima di averlo dimensionato, come abbiamo fatto noi all'inizio di questo paragrafo, il calcolatore lo dimensiona automaticamente a 10 per ogni dimensione. Un successivo dimensionamento produce ovviamente il messaggio REDIM'D ARRAY.

Le variabili con indice sono un elemento base della programmazione; esse consentono di utilizzare in pieno le possibilita' offerte dalle istruzioni per la gestione dei cicli

Gli indici possono essere numeri interi o reali, variabili numeriche intere o reali; di essi viene considerata solo la parte intera.

10.2 IL TRATTAMENTO DELLE STRINGHE

Un'interessante operazione che si puo' eseguire sulle variabili di tipo STRINGA e' quella del concatenamento: si puo' cioe' formare una stringa che contiene i caratteri di altre, due o piu', strighe, poste una dopo l'altra. L'operatore che concatena le stringhe e' il "+".
Vediamo un esempio:

```
A$="CIAO":B$="COMMODORE":C$=A$+B$
PRINT C$
```

Il calcolatore stampa il contenuto della stringa C$, cioe' CIAOCOMMODORE.

Non esistono particolari precauzioni da usare nel concatenare le stringhe: bisogna pero' fare attenzione affinche' la nuova stringa non superi i 255 caratteri, caso in cui viene emesso il messaggio d'errore STRING TOO LONG.
Il concatenamento delle stringhe deve essere usato per avere in memoria una stringa piu' lunga di 88 caratteri; infatti con l'istruzione INPUT non si puo' leggere un dato stringa che superi 88 caratteri.

Trattiamo ora le FUNZIONI del BASIC che lavorano sulle stringhe, cioe' quelle funzioni che hanno come argomento una stringa o che danno come risultato una stringa.

Le variabili di tipo stringa contengono una sequenza di caratteri, racchiusi dalle virgolette (SHIFT-2). I caratteri sono conservati nella memoria del calcolatore come numeri; ad ogni carattere corrisponde un codice numerico, il codice ASCII. Il codice del carattere A, ad esempio, e' 65. Il codice del carattere RETURN, e' 13. Tutti i caratteri che il calcolatore riconosce, anche quelli di controllo, hanno un codice. Nell'Appendice B sono descritti i codici di tutti i caratteri.

FUNZIONE ASC
La funzione fornisce un numero intero, che e' il valore del codice ASCII del primo carattere della stringa specificata; si scrive:
ASC(stringa), dove stringa puo' essere una costante tra virgolette o una variabile.
Se la stringa e' vuota, compare il messaggio d'errore ILLEGAL QUANTITY. Per evitarlo, in caso di stringhe, delle quali non e' sicuro il contenuto, puoi scrivere:
ASC(stringa+CHR$(0)).
Prova a digitare:
PRINT ASC("ABCD")
e il calcolatore stampa il numero 65, che e' il

carattere che compone la stringa ABCD.

FUNZIONE CHR$
Questa funzione si puo' considerare l'inverso della funzione ASC. Infatti essa ritorna una stringa, il cui codice ASCII e' quello della variabile o costante numerica compresa tra parentesi. Ad esempio puoi digitare:
PRINT CHR$(65)
e il calcolatore stampa una "A".
Questa funzione e' molto importante, poiche' permette di stampare tutti i caratteri, anche quelli che non e' possibile comprendere tra virgolette, come ad esempio ESC (vedi Paragrafo 1.4) e RETURN. L'argomento puo' variare tra 0 e 255 compresi. CHR$(0) equivale alla stringa nulla, ma non produce il messaggio di errore quando viene letto con la funzione ASC, come sopra indicato; PRINT ASC(""+CHR$(0)) stampa 0.

FUNZIONE LEN
Questa funzione ha come argomento una stringa, costante o variabile, e fornisce in uscita un valore intero, che ne esprime la lunghezza. Ad esempio, la stringa "CIAO" e' lunga 4 caratteri, e l'istruzione:
PRINT LEN ("CIAO")
stampa il numero 4.
Anche gli spazi bianchi sono considerati ovviamente caratteri, come anche tutti i caratteri di controllo e di punteggiatura.

FUNZIONE LEFT$
Questa funzione ha due argomenti: la stringa su cui deve operare e un numero intero. Left significa sinistra, e LEFT$ fornisce una stringa, che e' la parte sinistra della stringa specificata, della lunghezza espressa dal numero specificato. Esempio:
PRINT LEFT$ ("CIAO COMMODORE",7)
stampa CIAO CO, cioe' i primi 7 caratteri a par-

tire da sinistra della stringa "CIAO COMMODORE". Non esistono limitazioni sulla stringa: anche una stringa vuota e' accettata da questa funzione; l'unica limitazione sul numero intero e' che non deve essere negativo, e non deve superare 255.

FUNZIONE RIGHT$
Questa funzione e' sorella della precedente: right vuol dire destra, e RIGHT$ ritorna la parte destra della stringa specificata, della lunghezza espressa dal numero specificato. La sintassi e' la stessa di LEFT$. Esempio:
PRINT RIGHT$ ("CIAO COMMODORE",4)
stampa DORE, cioe' la parte destra della stringa "CIAO COMMODORE", della lunghezza di 4 caratteri. Le limitazioni sui parametri sono le stesse della funzione LEFT$.

FUNZIONE MID$
Questa funzione mette insieme le due precedenti, LEFT$ e RIGHT$. Essa lavora con 3 argomenti: una stringa e due numeri interi. La stringa e' quella su cui operare, e i due numeri indicano rispettivamente la posizione da cui partire (1=primo carattere da sinistra, 2=secondo, ecc.) e la lunghezza della nuova stringa. Esempio:
PRINT MID$("CIAO COMMODORE",9,4)
stampa MODO, che e' la stringa che parte dal nono carattere della stringa specificata, ed e' lunga 4 caratteri. La stringa, anche per MID$, puo' essere una qualunque stringa riconosciuta dal COMMODORE 16, il numero che indica la posizione di partenza deve essere compreso tra 1 e 255, il numero che esprime la lunghezza della stringa deve essere compreso tra 0 e 255.
Questa funzione puo' essere usata anche in altro modo, come pseudo variabile, e serve per modificare una parte di una stringa. Esempio:

```
A$="GATTO BRUTTO"
MID$(A$,7,6)="CARINO"
PRINT A$
```

stampa: GATTO CARINO
cioe' e' stato modificato il contenuto precedente di A$.

FUNZIONE INSTR
Questa funzione lavora su 2 parametri: due stringhe. La funzione cerca la seconda all'interno della prima stringa, se la trova, ritorna un numero, che indica a partire da quale posizione la seconda stringa e' contenuta nella prima. Se la seconda stringa non e' contenuta nella prima, il numero e' zero. Esempio:
PRINT INSTR("COMMODORE 16","MODO")
stampa 4, poiche' la stringa "MODO" parte dal quarto carattere della stringa "COMMODORE 16".
A questa funzione si puo' anche fornire un altro parametro, un numero che esprime da quale carattere partire nella ricerca. Vediamo un esempio:
PRINT INSTR("COMMODORE 16","O")
stampa 2.
PRINT INSTR("COMMODORE 16","O",3)
stampa 5, infatti la ricerca e' iniziata dal terzo carattere, e il quinto carattere e' una O. Come hai notato, la ricerca viene fatta da sinistra verso destra.

FUNZIONE VAL
Questa funzione serve per passare da una variabile di tipo stringa, a contenuto numerico, a una di tipo numerico: e' chiaro che il calcolatore rifiuta di applicare funzioni matematiche a variabili di tipo STRINGA, anche se queste contengono numeri. La funzione VAL ritorna un numero, che esprime il valore numerico della stringa specificata. Esempio:
PRINT VAL("16")+1
stampa 17, che e' 16+1. Vi sono alcune regole, riguardo a questa funzione, che bisogna conoscere:
. Se la stringa inizia con un carattere non numerico, il suo valore e' zero:

A=5:PRINT VAL ("A")
stampa 0 anche se la variabile A vale 5.
. La funzione VAL non effettua operazioni matematiche sulla stringa:
PRINT VAL("1+1")
non stampa 2, ma 1. VAL infatti tiene conto solo dei caratteri numerici, cosi' come sono, fino al primo carattere non numerico.
Sono considerate stringhe numeriche quelle contenenti un qualunque numero con segno, anche espresso in formato esponenziale.
Questa funzione e' utile negli INPUT dove si vuole ricevere un numero, per evitare che il BASIC invii un messaggio d'errore quando viene introdotta una stringa anziche' un numero:
INPUT A$:A=VAL(A$)
e' meglio di:INPUT A; la conversione si ferma al primo carattere non numerico incontrato.

FUNZIONE STR$
Questa funzione si puo' considerare l'inverso della funzione VAL, infatti riceve una variabile di tipo numerico, e ne ritorna una di tipo stringa. Puo' essere utile per applicare le funzioni LEFT$, MID$, RIGHT$, CHAR, alle variabili numeriche. Ad esempio:
CHAR 1,10,20,STR$(1234)
stampa la scritta 1234 in posizione 10,20, e non emette il messaggio TYPE MISMATCH, proprio perche' STR$ ritorna una variabile di tipo stringa.

10.3 LETTURA DATI DALL'INTERNO DEL PROGRAMMA

Il programma riceve i dati dall'esterno per mezzo dell'istruzione INPUT, oppure, carattere per carattere, con l'istruzione GET. A volte e' necessario incorporare in un programma blocchi di dati costanti; ogni singolo dato puo' essere assegnato a una variabile, ma devi scrivere molte istruzioni.

L'istruzione DATA si usa per inserire piu' comodamente dati nel programma. Per dati si intendono delle costanti, numeriche o stringhe, separate da virgole. Tutti i tipi di costanti usati dal COMMODORE 16 possono essere contenuti in linee DATA: numeri interi, reali in virgola fissa e in virgola mobile, stringhe. DATA e' un'istruzione che ha senso usare solo da programma, anche se in modo immediato non produce messaggio di errore.
I dati che sono archiviati in linee DATA vengono conservati nella memoria nell'ordine di comparizione delle relative linee DATA, che possono trovarsi ovunque in un programma. Per poter essere utilizzati essi devono essere trasferiti in variabili del programma, cioe' letti con l'istruzione READ. Esempio di alcune linee DATA:

```
1000 DATA 1,2,3,5,7,11,13,17,19:REM NUMERI PRIMI
1000 DATA A,E,I,O,U:REM VOCALI
1000 DATA "***   ***   ":REM SEGNI GRAFICI
1000 DATA1,A,2,B,3,C:REM UN NUMERO E UNA LET-
TERA
```

Anche per le linee DATA, come per la funzione VAL, non sono valide le espressioni matematiche. Le stringhe possono non essere comprese tra virgolette, a patto che non contengano caratteri di controllo o di interpunzione. Se invece sono comprese tra virgolette, possono contenere tutti i caratteri di controllo che e' consentito racchiudere tra virgolette. Una linea DATA, quando viene incontrata nel programma, viene saltata, come l'istruzione REM. A differenza di quest'ultima, pero', un successivo comando sulla stessa linea viene eseguito, anziche' essere ignorato. Esempio:

```
1000 DATA 1,2,3:PRINT "LINEA 1000"
```

stampa LINEA 1000 quando viene eseguito.
Naturalmente, a meno di non usare trucchi particolari, le linee DATA devono essere introdotte in fase di stesura del programma, e non possono essere introdotte in fase di esecuzione. Per

archiviare i dati che sono risultato di elaborazione non si puo' usare l'istruzione DATA, ma si deve ricorrere ai FILE (archivi) su nastro o su disco. La gestione degli archivi su disco e' abbastanza complessa, e sara' approfondita nel secondo volume. Se pero' conosci gia' la gestione dei FILE, puoi vedere nell'Appendice A qual'e' la sintassi dei comandi OPEN, PRINT#, GET#, INPUT#, CLOSE, e il loro uso, descritto sinteticamente.

L'istruzione READ e' sorella di DATA, infatti e' quella che permette di leggere i dati che sono archiviati in linee DATA. I dati vengono letti in sequenza, uno dopo l'altro. La prima volta che viene incontrata un'istruzione READ, viene letto il primo dato presente nella prima linea DATA. La seconda volta, viene letto il secondo dato, e cosi' via. L'istruzione READ deve essere seguita dal nome di una o piu' variabili (che concordino con il tipo delle costanti da leggere): ogni variabile, dopo l'esecuzione dell'istruzione, conterra' il valore letto dalle linee DATA. Prova il seguente esempio:

```
1000 DATA 1,A,2,B,3,C,4,D
```

e poi, in modo diretto, esegui per 5 volte la seguente linea:

```
READA,A$:PRINTA;A$
```

Come vedi, la prima volta il calcolatore stampa 1 A, cioe' i primi due dati; la seconda volta stampa 2 B (la seconda coppia di dati), e cosi' via. La quinta volta stampa OUT OF DATA ERROR. I dati infatti sono terminati.

Il calcolatore pone al primo dato (contenuto nelle istruzioni DATA) una specie di puntatore interno; esso avanza, dopo la lettura con READ, di un dato per ogni variabile letta.
L'istruzione RESTORE permette di riposizionare il puntatore o all'inizio del blocco di dati o a una determinata istruzione DATA del programma; cioe' consente di rileggere dati gia' letti o di

saltarne un gruppo.
Anche CLR ti permette di rileggere i dati dall'inizio, ma ti cancella anche tutte le variabili. RESTORE, puo' essere seguito da un numero di linea, che indica la linea dove si vuole che il prossimo dato venga letto. Prova il programma:

```
1000 DATA 1,2,3,4,5
1010 DATA 6,7,8,9,0
```

e introduci in immediato la linea seguente:

```
RESTORE 1010:READA,B:PRINTA;B
```

il calcolatore stampa 6 7, perche' ha letto i dati a partire dalla linea 1010.

10.4 L'ISTRUZIONE GET

L'istruzione GET serve per leggere dati dalla tastiera carattere per carattere; essa ha una particolarita', legge comunque quando viene eseguita, quindi non segnala che vuole un carattere, e, se non hai premuto alcun tasto, legge 0 se seguita da variabile numerica, stringa nulla, se seguita da variabile stringa.
Si scrive: GET variabile.
Essa puo' essere usata per leggere stringhe piu' lunghe di 88 caratteri (limite per INPUT), costruendo la stringa complessiva con l'operazione di concatenamento.
Per usarla correttamente si deve garantirsi dalla lettura di niente operando cosi':

```
10 A$="":GET A$:IF A$="" THEN 10
```

con questa linea di programma si prosegue solo se si preme un tasto; puoi aggiungere:

```
20 PRINT A$
```

e fare delle prove.

Un tipico uso di questa istruzione e' quello di creare un ciclo di attesa fino alla pressione di un particolare tasto:

```
500 A$="":GET A$:IF A$<>"P" THEN 500
```

prosegue solo se premi P.

Esiste un'altra istruzione la GETKEY, che fa proseguire solo se si preme un tasto; si scrive:

GETKEY lista variabili

per esempio:

600 GETKEY A$

non prosegue fino a quando non si preme un tasto. I tasti funzione danno errore.

10.6 RIEPILOGO

In questo capitolo ci siamo occupati di:

.VARIABILI CON INDICE
.CONCATENAMENTO DELLE STRINGHE
.FUNZIONI CHE LAVORANO SU STRINGHE
.GESTIONE DATI ALL'INTERNO DEL PROGRAMMA
.ISTRUZIONI: GET E GETKEY

se non hai le idee chiare rivedi il programma CAP10 e dopo rileggi questo capitolo.

CAPITOLO 11

OPERAZIONI AVANZATE

11.1 LE FINESTRE SUL VIDEO

La finestra video e' una particolarita' che ti permette di definire una porzione di video dove lavorare: la zona al di fuori di questa porzione non viene interessata ne' da nuove scritte, ne' dall'operazione dello SCROLLING. Praticamente puoi decidere tu quante devono essere le righe e quante le colonne su cui scrivere, a patto di non superare le 25 righe e le 40 colonne, facendo in modo che tutto il resto dello schermo non sia alterato. Il programma ES11.1 ti mostra un'applicazione della finestra video: una parte dello schermo viene riservata a una scritta fissa, e il resto viene lasciato a tua disposizione per usare il calcolatore. Come puoi vedere, la scritta lampeggiante non va via, neppure se premi il tasto SHIFT-CLR/HOME (vedi Paragrafo 1.4).

```
0 REM ES11.1
10 PRINTCHR$(147);
20 PRINT"┌──────────────────────";
25 PRINT"──────────────────────┐"
30 PRINT"|                    "CHR$(130)"COMMODORE 16";
35 PRINTCHR$(132)"                  |"
40 PRINT"└──────────────────────";
45 PRINT"──────────────────────┘"
50 CHAR,0,3,CHR$(27)+"T"
```

COME INSERIRE LA FINESTRA VIDEO

Abbiamo accennato nel Paragrafo 1.4 che la sequenza ESC T posiziona l'angolo in alto a sinistra della finestra video, mentre la sequenza ESC B ne posiziona l'angolo in basso a destra. Per posizionare l'angolo in alto a sinistra quindi e' sufficiente portare il cursore nel punto in cui lo si vuole posizionare, e premere i tasti ESC e poi T. In questo modo abbiamo posizionato uno dei due parametri che occorrono per definire la nostra finestra video: l'inizio. Per l'altro (la fine della finestra), ci posizioneremo col cursore dove abbiamo stabilito che si deve trovare l'angolo in basso a destra e premeremo ESC e poi B. Da questo momento non potremo piu' scrivere fuori dalla zona delimitata, se non eliminando l'effetto della finestra.

Per inserire la finestra da programma, bisogna portare il cursore nell'angolo della finestra, stampando i caratteri di controllo necessari, poi stampare CHR$(27) (che e' il codice ASCII di ESC) e "T" o "B" a seconda che si tratti dell'angolo iniziale, in alto a sinistra, o di quello finale, in basso a destra.

COME TOGLIERE LA FINESTRA VIDEO

Per eliminare la finestra, possiamo premere due volte di fila il tasto CLR/HOME. Ovviamente anche premendo il tasto RESET la finestra video si cancella, ma cosi' si cancella anche il programma contenuto in memoria; invece, RUN/STOP-RESET cancella la finestra video, senza cancellare il programma in memoria (vedi Paragrafo 1.6). Un altro modo per cancellare la finestra video e' premere i tasti ESC N; cosi' pero' tutto quello che si trova sullo schermo viene cancellato. ESC R produce automaticamente una finestra video di 23 righe X 38 colonne, e cancella tutto lo schermo.

Per togliere la finestra da programma, basta far esegire l'istruzione:

PRINT CHR$(19) CHR$(19)
che equivale a premere due volte di fila il tasto CLR/HOME.

11.2 DEFINIZIONE DELLE FUNZIONI UTENTE

ISTRUZIONE DEF FN
DEF FN significa DEFINE FUNCTION (definisci la funzione). Per funzione si intende una funzione matematica, cioe' una qualsiasi relazione che riceve in ingresso un valore e ne ritorna un altro. DEF FN va seguito dal nome della funzione e, tra parentesi, dal nome della variabile, su cui deve lavorare. La variabile su cui lavora la funzione si chiama ARGOMENTO. Ecco un esempio:

```
10 DEF FN F(X)=X*X
```

Abbiamo definito una funzione che eleva al quadrato l'argomento. Se infatti chiediamo il valore della funzione, aggiungendo la linea:

```
20 PRINT"-4 AL QUADRATO FA" FN F(-4)
```

e scriviamo RUN, vediamo che il calcolatore stampa:

```
-4 AL QUADRATO FA 16
```

Come hai notato, la X non e' una vera variabile; essa serve solo per definire la funzione. Quando la funzione viene richiamata, i calcoli sono eseguiti sull'argomento che viene passato in quel momento. Nel nostro esempio e' stata passata come argomento la costante -4.
La X contenuta tra parentesi nella definizione della funzione non ha relazione con una eventuale variabile X presente in altre parti del programma.
Nel COMMODORE 16 si puo' passare solo un argomento alla funzione: NON si puo' cioe' usare una funzione di piu' variabili del tipo:

```
DEF FN F(X,Y)=SQR(X*X+Y*Y)
```

ma bisogna passare una variabile solamente, tra le parentesi.

```
10 DEF FN F(X)=SQR(X*X+Y*Y)
```

In questo caso, potremo calcolare la funzione

nel modo seguente:

```
20 Y=4:PRINT FN F(3)
```

In questo modo abbiamo calcolato il valore di una funzione di due variabili, passandone una tra parentesi, e l'altra attraverso la variabile Y. Il risultato di questo programma e' stampare 5, che e' l'ipotenusa di un triangolo rettangolo coi cateti che valgono 3 e 4.
Ricorda di porre la linea ove definisci la funzione all'inizio nel programma, perche' viene emesso il messaggio UNDEF'D FUNCTION (funzione non definita) se viene richiesto il valore di una funzione di cui non e' ancora stata incontrata la definizione:

```
10 PRINT FN Q(3)
20 DEFFN Q(Z)=12
RUN
```

Il calcolatore si arresta con il messaggio d'errore UNDEF'D FUNCTION ERROR IN 10 perche' il programma ha trovato prima un'istruzione FN di una DEF FN. Scambiando il numero delle linee il programma funziona correttamente.

11.3 ALCUNE FUNZIONI AVANZATE

LA FUNZIONE RND(X)

Questa funzione ritorna un numero REALE pseudo CASUALE compreso tra 0 e 1 escluso. Per ottenere un numero compreso tra X e Y, bisogna moltiplicare il numero casuale per (Y-X) e sommare X. Se poi si vuole che il numero sia intero, si puo' usare la funzione INT. Scriviamo una linea che ritorni un numero intero casuale compreso tra 1 e 7, escluso 7:

```
PRINT INT(RND(1)*6+1)
```

Stampa un numero intero casuale compreso tra 1 e 7, escluso 7, cioe' tra 1 e 6 compresi.
Generare numeri casuali e' un problema abbastanza complicato, per una macchina precisa come un calcolatore. Occorre infatti avere una base

casuale da cui partire. Nel COMMODORE 16 vi sono tre modi per ottenere le basi dei numeri casuali, che si selezionano scegliendo il valore dell'argomento della funzione:
-RND(0) produce il numero casuale partendo da un contatore interno al calcolatore, che viene continuamente e rapidamente incrementato: e' il modo nel quale il numero piu' si avvicina a un vero numero casuale
-RND(numero negativo) genera il numero casuale partendo dal numero tra parentesi:

PRINT RND(-1)
stampa sempre 2.99196472E-08
-RND(numero positivo) genera il nuovo numero partendo dal precedente numero casuale generato, permettendo di generare una serie di numeri casuale ripetibile:
PRINT RND(1), appena acceso il calcolatore, stampa 1.07870447E-03. La seconda volta stampa .793262171. Anche RND(4) stampa lo stesso risultato di RND(1), perche' il seme non dipende dal valore dell'argomento della funzione, ma solo dal suo segno.

LA FUNZIONE PEEK(X)
E' una funzione che dovrebbe essere usata da un programmatore che conosce a fondo la macchina. Ritorna il valore del byte di memoria di indirizzo X. X non puo' essere negativo, ne' maggiore di 65535, perche' il COMMODORE 16 puo' indirizzare 65536 byte diversi (tra 0 e 65535).

IL COMANDO POKE X,Y
E' il contrario di PEEK, cioe' anziche' leggere il contenuto del byte X, vi scrive il numero Y. Y deve essere compreso tra 0 e 255, e per il valore di X vedi PEEK.
POKE 16384,200:PRINT PEEK(16384)
stampa 200. Anche premendo NEW, CLR/HOME, il contenuto di questo byte non cambia. Poiche' la memoria viene usata dal calcolatore per molte

funzioni, cambiare il contenuto di byte di memoria di cui non si conosce l'uso e' pericoloso, in quanto puo' facilmente causare l'arresto completo del calcolatore.

11.4 COME TROVARE GLI ERRORI NEL PROGRAMMA

Il COMMODORE 16 possiede diversi comandi che ti aiutano nel "debugging", cioe' la correzione, del programma.
Questi comandi sono HELP, TRON, TROFF, STOP. A volte ti puo' aiutare anche l'Appendice E, che riporta la maggior parte delle cause che procurano gli errori solitamente piu' "difficili" da scoprire.

IL COMANDO HELP

E' forse il comando piu' potente a disposizione per capire qual'e' l'istruzione che ha prodotto l'errore. Puoi ottenere di eseguire questo comando premendo il tasto di funzione con la scritta HELP, o digitando il comando da tastiera, dopo che si e' verificato un errore nel programma. Vedrai lampeggiare un pezzo di linea. La prima istruzione che lampeggia e' quella nella quale si e' verificato l'errore, quella cioe' che non e' stata portata a termine a causa dell'errore. HELP non funziona sui comandi immediati. Purtoppo capita anche ai programmatori piu' esperti di fare errori nei programmi. Le tecniche moderne di programmazione insegnano a scrivere programmi molto ordinati, in modo da dividere il programma in MODULI (parti il piu' possibile indipendenti). Tale tecnica permette di PROVARE separatamente ogni MODULO in tutte le condizioni piu' "critiche" per ciascun modulo.
Cerca di rendere i programmi ordinati e leggibili: risparmierai tempo in fase di correzione e di modifica.

IL COMANDO TRON
TRON vuol dire TRACE ON, cioe' "segna" i numeri di tutte le linee che esegui. In questo modo e' possibile avere una traccia di dove il programma e' passato, e capire qual'e' la causa di un malfunzionamento.

IL COMANDO TROFF
E' il contrario di TRON. Permette di evitare che il calcolatore continui a stampare i numeri di linea delle linee eseguite anche dopo che non occorre piu' stampare la traccia del programma.

IL COMANDO STOP
Questo comando permette di arrestare il programma in determinati punti "critici", per richiedere il valore di variabili importanti, o per vedere se il programma passa in certi punti. E' possibile riprendere l'esecuzione del programma dopo uno STOP, col comando CONT, a patto di non aver modificato il programma, di non avere commesso errori in modo immediato e di non aver eseguito istruzioni che distruggano il programma o le variabili (NEW, CLR, ecc.).

11.5 LA GESTIONE DEGLI ERRORI NEL PROGRAMMA

Il COMMODORE 16 ti da' la possibilita' di scrivere una routine di gestione degli errori, una routine cioe' che viene eseguita automaticamente quando viene incontrato un errore. E' ovvio che una routine che deve gestire gli errori e' un po' "delicata", nel senso che puo' produrre un ciclo senza fine e inarrestabile, se gestita male: prova ad esempio il seguente programma":

```
10 TRAP 100
20 GOTO 20
100 RESUME
```

Puoi arrestare questo programma solo premendo il tasto di RESET.

IL COMANDO TRAP

TRAP e' il comando che permette di definire il numero di linea dove parte la routine di gestione degli errori. L'istruzione:

TRAP 20000

informa il calcolatore che in caso di errore deve andare alla linea 20000 anziche' emettere il messaggio e arrestare l'esecuzione. Usato da solo, cioe' senza il numero di linea, TRAP toglie l'effetto "trappola" e ritorna nel modo normale, in cui in caso di errore viene emesso il messaggio di errore del sistema e viene arrestata l'esecuzione. In questo modo e' possibile inserire e disinserire l'effetto trappola lungo il programma, in modo da utilizzarlo solo in quei punti dove e' prevedibile un errore. Tra tutti i messaggi di errore ce n'e' uno solo che non "sente" l'effetto trappola: UNDEF'D STATEMENT. Del resto in un programma corretto tale situazione non si dovrebbe mai verificare. TRAP inoltre si puo' usare solo in modo programma, e non in modo diretto.

LA ROUTINE DI GESTIONE DEGLI ERRORI

Abbiamo accennato che questa routine e' delicata: ricorda di gestire solo gli errori che hai previsto, e di non riprendere l'esecuzione del programma se l'errore non e' quello che avevi previsto. Vi sono nel COMMODORE 16 due variabili riservate, ER e EL, che indicano rispettivamente il numero di codice dell'errore e il numero della linea dove si e' verificato l'errore. Vi e' anche una funzione, ERR$(X), che ritorna una stringa contenente il messaggio di errore dell'errore che ha codice X. Scriviamo una ruotine degli errori che non ti permette di arrestare il programma col tasto RUN/STOP:

```
20000 IF ER <> 30 THEN PRINT ERR$(ER) "ERROR IN"
EL: END
20010 PRINT "NON PUOI ARRESTARE IL PROGRAMMA SE
NON CON RESET"
20020 RESUME
```

Una routine di errore cosi' non permette di arrestare il programma col tasto STOP (codice di errore = 30), ma stampa il messaggio di errore se si verifica qualunque altro errore. Il COMMODORE 16 e' costruito in modo da non gestire, se non in modo sistema, eventuali errori che si verificano nella routine di gestione degli errori: in tale circostanza viene emesso il messaggio di errore relativo all'ultimo errore verificatosi. Come probabilmente hai notato, tale routine assomiglia molto a una subroutine, che quindi non deve terminare con GOTO. A diffetenza della subroutine, non deve terminare con RETURN, ma SEMPRE con RESUME.

IL COMANDO RESUME
Va sempre usato per terminare una routine di gestione degli errori, altrimenti il calcolatore ritiene di trovarsi ancora in tale routine, e non esegue piu' la routine degli errori se si verifica un altro errore. Il calcolatore ENTRA in modo GESTIONE ERRORE quando si verifica un errore, e ne esce solo quando incontra RESUME. Vi sono tre diversi modi di usare RESUME:
-RESUME da solo riprende l'esecuzione del programma principale ripetendo l'istruzione nella quale si e' verificato l'errore
-RESUME NEXT riprende dall'istruzione successiva a quella dove si e' verificato l'errore
-RESUME numero linea riprende a partire dal numero di linea specificato.

11.6 GESTIONE FILE DI DATI SU CASSETTA

Si chiama FILE un insieme di registrazioni; un programma registrato su cassetta e' un file.
La registrazione di un file puo' essere eseguita su un qualunque supporto fisico; puo' essere considerato un file anche uno stampato su carta.

Ci occupiamo qui dei file di dati registrati su cassetta magnetica, e introduciamo i concetti necessari alla comprensione dell'argomento. Questi concetti sono in generale validi anche per i file registrati su altri supporti fisici.
La natura del supporto fisico influisce sulle possibilita' di registrazione; nel caso della cassetta e' possibile registrare facendo scorrere il nastro in sequenza, si tratta di registrazione sequenziale. Ovviamente se una registrazione e' di tipo sequenziale, anche la conseguente rilettura puo' essere solo di tipo sequenziale.

In molti casi e' possibile raggruppare le registrazioni, in modo tale che si possano reperire in sequenza le informazioni relative a un determinato soggetto. Per esempio, in una scuola raggruppiamo i dati relativi a ogni studente e chiamiamo RECORD ogni singolo gruppo di registrazioni. Continuando con il nostro esempio, se la scuola ha 500 studenti, esiste un FILE di nome STUDENTI, con 500 RECORD, uno per ogni studente. All'interno del record le singole informazioni si chiamano CAMPI (FIELD) e ogni campo puo' essere di tipo diverso, come cognome, nome, data di nascita, luogo di nascita, data di iscrizione, numero di matricola, ecc.

Possiamo schematizzare il nostro file, in un pezzo di nastro magnetico, sul quale si susseguono i record, ognuno diviso in campi, come in Figura 11.1.

In generale fa parte del SISTEMA OPERATIVO del calcolatore il software per la gestione dei file; si tratta di un gruppo di subroutine che gestiscono la trasmissione dei dati con la periferica in questione. Dipende sia dalla periferica che dal software di gestione quello che e' consentito fare.

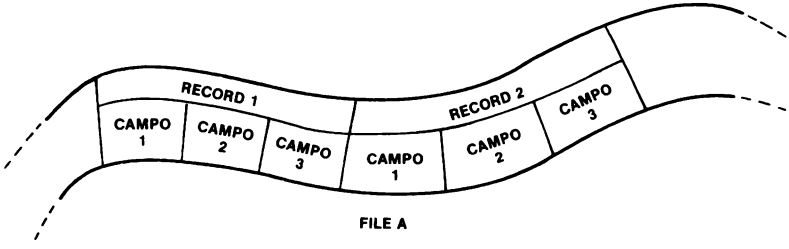


Figura 11.1 File, record e campi

Nel caso della cassetta collegata al COMMODORE 16 e' possibile gestire FILE SEQUENZIALI di dati, nei quali i campi sono registrati uno dopo l'altro. E' possibile leggere in sequenza il contenuto di un file partendo dalla prima registrazione. Un file puo' essere o LETTO o SCRITTO, non si possono mescolare le due operazioni.

Non esistono limiti alla lunghezza dei singoli campi (salvo le limitazioni gia' note sulla grandezza dei numeri e la lunghezza delle stringhe).
Risulta abbastanza laboriosa l'operazione di aggiornamento di un file; infatti non si puo' correggere una parte di registrazioni. Per aggiornare un file su cassetta e' necessario trasferire il suo contenuto in memoria, e poi scrivere un nuovo file, modificando dove necessario. In conseguenza ti raccomandiamo di non creare file di dati troppo lunghi; devi sempre fare i conti con la memoria disponibile nel calcolatore.
Se vuoi organizzare un archivio di indirizzi, per esempio, invece di creare un unico file di nominativi, puoi spezzarlo in una serie di file, ognuno dedicato ai nomi che iniziano con un gruppo di lettere.

Le operazioni per la gestione dei file sono:

.1) apertura della comunicazione con il file,
.2) scrittura o lettura del file,
.3) chiusura della comunicazione con il file.

Al momento della creazione di un file, gli assegnamo un nome; questo nome va ricordato per poter comunicare con il file e viene registrato all'inizio del file.
La periferica e' contraddistinta da un numero che chiamiamo "dn" da Device Number; per la cassetta abbiamo dn=1.
Il file e' visto dal programma, oltre che con un nome, anche con un numero distintivo, che chiamiamo "lfn", da Logical File Number; esso puo' variare da 1 a 255 (di norma si usano numeri piccoli).
Dobbiamo considerare un ultimo parametro che interviene nelle istruzioni per la gestione dei file; esso e' "sa" da Secondary Address, e serve per precisare il tipo dell'operazione. Nel seguito troveremo:
.fn, nome file, stringa da 1 a 16 caratteri,
.lfn, numero logico del file, da 1 a 255,
.dn, numero della periferica, 1 per la cassetta,
.sa, indirizzo secondario, che per la cassetta puo' valere:
..0 o mancare per operazioni di lettura da nastro,
..1 per operazioni di scrittura su nastro, con registrazione del carattere di fine file,
..2 per operazioni di scrittura su nastro con registrazione anche di un altro carattere particolare alla fine del file, per segnalare anche la fine del nastro.
I parametri dn, lfn e sa, possono essere costanti o variabili numeriche; fn puo' essere una costante o una variabile stringa.

Per stabilire la comunicazione con un file si

usa l'istruzione OPEN; essa si scrive:

OPEN lfn,dn,sa,fn

dove i parametri hanno il significato visto sopra. Con la OPEN viene assegnato al file di nome "fn" il numero logico "lfn"; tale file viene aperto, cioe' dopo la OPEN si puo' leggere o scrivere. Il parametro "dn" deve essere 1 per la cassetta; il tipo di operazione consentito sul file dopo l'apertura dipende dal valore di "sa". Se il file e' stato aperto per lettura, e quindi deve esistere gia' sul nastro, quando tenti di scriverci sopra ottieni un messaggio di errore.
In conseguenza del valore di "sa" al momento dell'esecuzione della OPEN compare sul video il messaggio di richiesta:
PRESS PLAY ON TAPE, oppure
PRESS PLAY & RECORD ON TAPE.
Quando viene eseguita la OPEN, dopo la risposta al messaggio di sistema, il video si sbianca, il nastro gira, e, in caso di lettura viene ricercata l'intestazione del file richiesto, mentre in caso di scrittura viene scritta la testata del nuovo file.

Le altre istruzioni per la gestione del file si riferiscono al file aperto per mezzo del numero "lfn".

Per scrivere si usa l'istruzione PRINT#; essa si scrive:

PRINT#lfn, lista dati

Scrive sul file aperto con numero logico "lfn" i dati della lista, che possono essere variabili o costanti. In particolare lista dati puo' essere una sola variabile.
E' importante sottolineare che i dati vengono registrati carattere per carattere sul nastro, e

che e' necessario registrare un "carattere separatore" che, in fase di lettura, segnali la fine di un dato. Tale carattere separatore deve essere o la virgola o il RETURN. Il carattere separatore virgola puo' essere passato come: "," oppure CHR$(44), mentre il RETURN puo' essere passato come CHR$(13), oppure dall'assenza di punteggiatura dopo l'ultimo dato della lista. Esempi:

```
10 PRINT#3,"alba";"serena"
```

da' luogo in lettura a un solo dato che appare come "albaserena", infatti il separatore punto e virgola non ha fatto aggiungere spazi e non e' presente un separatore registrabile, mentre la mancanza di punteggiatura finale ha fatto registrare un RETURN.

```
10 PRINT#3,"alba","serena"
```

da' luogo in lettura a un solo dato che appare come "alba serena", infatti il separatore virgola ha fatto aggiungere 8 spazi.

```
10 PRINT#3,"alba";",";"serena"
```

da' luogo in lettura a due dati: il primo "alba" e il secondo "serena", infatti e' stata registrata la virgola separatrice. Tali dati pero' possono essere letti tutti e due solo da un'istruzione di lettura che contenga due variabili nella lista. Infatti una operazione di lettura legge sempre tutti i dati compresi tra due caratteri RETURN; se non poni un numero sufficiente di variabili nella lista di lettura puoi perdere dei dati. Questo non succede se il separatore e' il RETURN per ogni singolo dato.

```
10 PRINT#3,"alba":PRINT#3,"serena"
```

da' luogo in lettura a due dati, che sono stati

registrati con il carattere separatore RETURN, infatti manca la punteggiatura alla fine della lista dati.

Durante le operazioni di scrittura di un file non avviene un reale trasferimento di dati tra calcolatore e cassetta ogni volta che si esegue una PRINT#; l'operazione di trasferimento dati avviene solo quando, dopo l'esecuzione di un certo numero di istruzioni PRINT#, e' stata riempita la zona di memoria che serve per immagazzinare i dati per la cassetta; tale zona si chiama BUFFER e puo' contenere 192 caratteri. Quando il buffer e' pieno, il suo contenuto viene trasferito sulla cassetta; durante il trasferimento il video si sbianca.

Per leggere un file si usa l'istruzione INPUT#; essa si scrive:

INPUT#lfn,lista dati

e lista dati deve contenere i nomi delle variabili nelle quali leggere da nastro, separate da virgole. La INPUT# legge i dati che sono registrati tra due RETURN, perdendone alcuni, se il numero di variabili della lista non e' sufficiente. Inoltre il numero dei caratteri tra due RETURN non deve superare 88.
Il tipo delle variabili deve concordare con il tipo dei dati da leggere; in caso contrario si ha segnalazione di errore.

Per poter leggere con maggior liberta', si puo' usare l'istruzione GET#, che legge carattere per carattere, compresi i caratteri separatori registrati; essa si scrive:

GET#lfn,lista dati

E' consigliabile usare nomi di variabili stringa per evitare segnalazioni di errore.

Con GET# si possono leggere campi lunghi a piacere, dato che si leggono un carattere per volta.

Anche per le operazioni di lettura il trasferimento dei dati avviene a blocchi; alla prima lettura viene riempito il buffer e vengono poi prelevati da esso i caratteri. Il buffer viene riempito nuovamente quando tutti i dati presenti sono stati letti.

Per chiudere la comunicazione con un file si usa l'istruzione CLOSE; essa si scrive:

CLOSE lfn

ed e' necessario usarla per lavorare correttamente. Chiudere un file, aperto per scrivere, e' necessario anche per svuotare il buffer di eventuali caratteri residui. L'operazione di chiusura serve anche per liberare una posizione nella tabella di gestione dei file del BASIC; tale tabella consente di gestire solo 10 file contemporaneamente.

Riportiamo come esempio di scrittura di un file il programma CREAFILE.

```
1 REM CREAFILE
10 OPEN 3,1,1,"NOMI"
15 FORK=1TO10
20 INPUT"NOME=";N$
25 PRINT#3,N$
30 NEXTK
40 CLOSE 3
```

In esso alla linea 10 viene aperto il file NOMI per scrivere e gli viene assegnato lfn=3. Dalla linea 15 alla 30 si ha un ciclo nel quale vengono chiesti 10 nomi dalla tastiera e scritti sul nastro. Alla linea 40 viene chiuso il file.

Segue il programma esempio LEGGIFILE, nel quale viene aperto in lettura il file NOMI, vengono letti in ciclo e stampati sul video i 10 nomi e poi viene chiuso il file.

```
1 REM LEGGIFILE
10 OPEN 3,1,0,"NOMI"
15 FORK=1TO10
20 INPUT#3,N$
25 PRINT N$
30 NEXTK
40 CLOSE 3
```

In quest'ultimo esempio abbiamo letto 10 dati, infatti sapevamo di averne scritti 10. Se avessimo cercato di leggerne piu' di 10 avremmo avuto segnalazione di errore. Quando il file, aperto in scrittura, viene chiuso, il sistema aggiunge in fondo un carattere particolare che segnala la fine del file e si chiama EOF (da End Of File); anche il carattere EOT (da End Of Tape) se sa=2. Il carattere EOF, che e' uno 0 (codice ASCII 0) viene sentito quando si legge l'ultimo dato che lo precede, e viene registrato automaticamente dal sistema nella variabile riservata ST, che viene posta a 64. Per questa ragione dopo ogni lettura e' bene memorizzare ST in una variabile di comodo, da analizzare prima di tornare a leggere. Il carattere EOT viene prodotto da te se scrivi su un file il carattere CHR$(0); quindi devi evitare di usarlo, dato che il calcolatore lo riconosce come segnalazione di fine file. Il programma LEGGIFILE, e' meglio sia scritto come LEGGICONST, che segue.

```
1 REM LEGGICONST
10 OPEN#3,1,0,"NOMI"
15 INPUT#3,N$
20 TS=ST
25 PRINTN$
30 IFTS=0THEN15
40 CLOSE3
```

In esso, dopo l'istruzione INPUT#3, memorizziamo in TS la variabile di stato ST. Alla linea 30 analizziamo TS e torniamo a leggere dal file solo se essa e' a zero, altrimenti chiudiamo il file.

Nel caso dei file su cassetta non sono applicabili in modo rigoroso i concetti di RECORD e di CAMPO. Infatti potresti considerare RECORD la parte di registrazioni comprese tra due RETURN, e CAMPI i dati, separati da virgola, all'interno del record, ma avresti le seguenti limitazioni:
. il RECORD non potrebbe contenere piu' di 88 caratteri,
. il RECORD dovrebbe essere sempre letto da una sola INPUT#, usando un numero di variabili pari al numero dei CAMPI, oppure dovresti usare GET# e leggere carattere per carattere.
Per tutte queste ragioni, in generale i file sequenziali su cassetta si registrano usando il RETURN (CHR$(13)) sia come separatore di campo che come separatore di record.

11.7 ESEMPIO ARCHIVIO DI DATI SU CASSETTA

Riportiamo, come esempio riassuntivo, un programma che consente di gestire un archivio di dati su cassetta. Costruiamo il programma in modo che sia facilmente modificabile, e tu lo possa adattare alle tue esigenze, cambiando sia i nomi dei campi, che il loro numero.

Le operazioni che deve essere possibile effettuare per gestire un archivio di dati sono le seguenti:

.a) creazione ex novo dell'archivio,
.b) aggiornamento dell'archivio:
   b1) cancellazioni di record,
   b2) aggiunta di nuovi record,
   b3) modifica di record gia' esistenti,

.c) lista di tutto o parte dell'archivio.
Queste operazioni sono in generale necessarie per qualunque archivio; inoltre ogni archivio puo' essere utilizzato per altri scopi, che dipendono dalle specifiche esigenze inerenti ai soggetti archiviati.

Nella Figura 11.2 riportiamo uno schema a blocchi molto generale del programma FILESEQ, per la gestione di un archivio sequenziale.

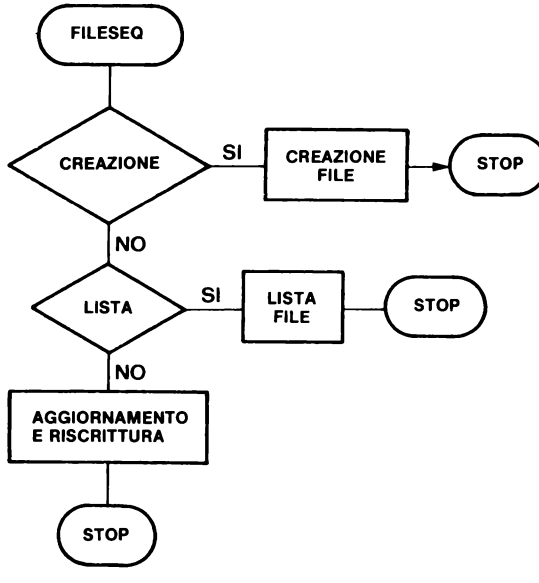


Figura 11.2 Diagramma a blocchi FILESEQ

Noi abbiamo deciso di creare un archivio formato da record di 6 campi ciascuno, relativi a un'agenda di indirizzi. Creiamo e manteniamo l'archivio in ordine alfabetico in base ai due primi campi; in conseguenza i dati, nella fase

di creazione iniziale del file devono essere forniti rispettando tale ordine. Il programma rifiuta nominativi fuori ordine, come pure nominativi uguali.
All'inizio viene proposto un menu' per scegliere tra le tre funzioni principali: CREAZIONE, AGGIORNAMENTO, STAMPA. Viene chiesto quanti record si vogliono trattare, per predisporre il dimensionamento delle variabili.
La funzione di AGGIORNAMENTO inizia trasferendo completamente in memoria il file esistente; poi si articola in tre fasi:
.1) modifica in memoria dei record gia' esistenti,
.2) cancellazione in memoria, ponendo spazi al posto del primo campo, nei record da eliminare dal file,
.3) aggiunta, durante la riscrittura del file, di eventuali nuovi record, che devono essere forniti in ordine.

In Figura 11.3 riportiamo uno schema a blocchi della fase di aggiornamento.

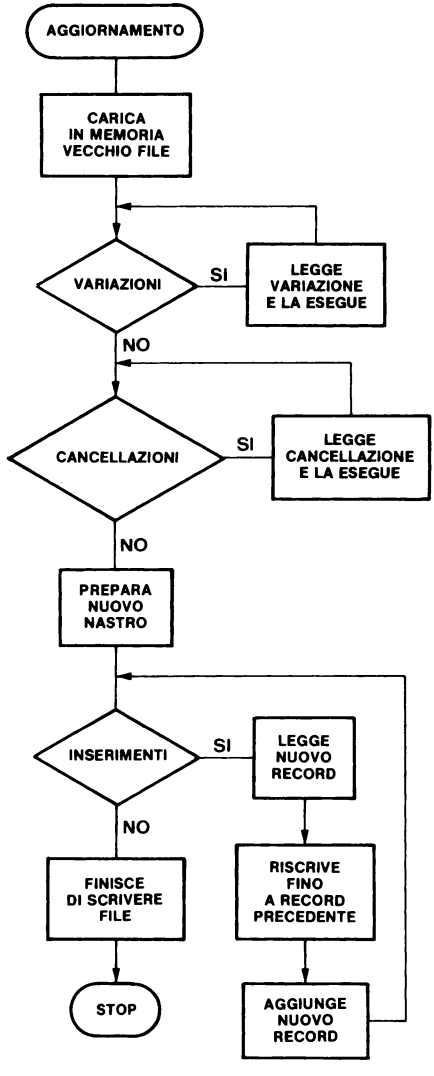


Figura 11.3 Fase AGGIORNAMENTO di FILESEQ

Riportiamo il listato del programma FILESEQ, che trovi anche memorizzato sulla cassetta.

```
1 REM FILESEQ
3 REM CREAZIONE E GESTIONE FILE SEQUENZIALE
5 REM RECORD IN ORDINE PRIMI DUE CAMPI
7 PR=4:REM DN PERIFERICA DI STAMPA
9 NF=4:REM NUMERO LOGICO FILE DI STAMPA
11 NC=6:REM NUMERO CAMPI DEL RECORD
13 REM VETTORI DESCRIZIONI E DATI
15 DIMD$(NC),I$(NC)
17 CH$=CHR$(13):SP$="   "
19 REM DESCRIZIONI CAMPI
21 DATA COGNOME, NOME,INDIRIZZO
23 DATA CAP,CITTA',TEL.
25 FORK=1TONC:READD$(K):NEXTK
27 REM SCELTA OPERAZIONI
29 PRINT"[▼][▼]           SCELTA OPERAZIONE[▲][▲]"
31 PRINTTAB(10)"1) CREAZIONE"
33 PRINTTAB(10)"2) AGGIORNAMENTO"
35 PRINTTAB(10)"3) STAMPA"
37 PRINTTAB(10)"9) FINE[▲]"
39 GETKEYR$:R=VAL(R$)
41 IFR<1OR(R>3ANDR<>9)THEN39
43 IFR=9THENGOSUB365:STOP
45 PRINT"[▼][▼][▼][▼]    QUANTI RECORD TRATTI IN TUTTO,"
47 PRINT"    SE IL FILE NON ESISTE 0,"
49 PRINT"    SE IL FILE ESISTE GIA'?"
51 INPUT"[▲]NUMERO RECORD: ";N
53 INPUT"CONFERMI (S/N): ";R$
55 IFR$<>"S"THEN45
57 REM DIMENSIONAMENTO MEMORIA PER N RECORD
59 DIMMC$(N,NC)
61 ON R GOTO 63,113,91
63 REM
65 REM CREAZIONE FILE
67 GOSUB267:GOSUB275:GOSUB281
69 LC$="":LN$="":K1=N:FORK=1TON
71 GOSUB211
73 IFSW<>0THENK1=K-1:K=N:GOTO85
75 IFI$(1)>LC$THEN81
```

```
77 IFI$(1)=LC$THENIFI$(2)>LN$THEN81
79 GOSUB313:GOSUB319:GOTO71
81 LC$=I$(1):LN$=I$(2)
83 GOSUB221
85 NEXTK:CLOSE1
87 PRINT"▽FINITO CARICAMENTO   ";K1;" RECORD"
89 PRINT"▽FILE: ";NF$:GOSUB319:GOSUB365:RUN
91 REM
93 REM STAMPA FILE
95 GOSUB285:GOSUB267:GOSUB275:GOSUB293
97 PRINT#NF,"LISTA FILE ";NF$:PRINT#NF:PRINT#NF
99 GOSUB299
101 PRINT#NF,I$(1);SP$;I$(2)
103 PRINT#NF,I$(3);SP$;I$(4);SP$;I$(5);SP$;I$(6)
105 PRINT#NF:PRINT#NF
107 IFF$<>64THEN99
109 CLOSE1:CLOSENF:PRINT"▽▽FINITO LISTA"
111 GOSUB365:RUN
113 REM
115 REM AGGIORNAMENTO FILE
117 PRINT"▽▽▽AGGIORNAMENTO FILE"
119 PRINT"▽▽MONTA NASTRO VECCHIO"
121 GOSUB319:GOSUB275:GOSUB291:FORK=1TON
123 FORJ=1TONC:INPUT#1,MC$(K,J):NEXTJ
125 IFST=64THENCLOSE1:K1=K:K=N:NEXTK:GOTO129
127 NEXTK:PRINT"▽FILE SUPERA NUM.MASS.REC.";N:STOP
129 PRINT"▽LETTI ";K1;" RECORD":SW=0:N=K1
131 PRINT"▽▽VARIAZIONI S/N ":INPUTR$
133 IFR$<>"S"THEN149
135 REM
137 REM MODIFICA RECORD
139 SW=0:GOSUB375:IFSW<>0THEN149
141 SW=0:GOSUB387:IFSW<>0THEN135
143 FORJ=3TONC:I$(J)=MC$(K1,J):NEXTJ
145 GOSUB357:GOSUB337
147 FORJ=3TONC:MC$(K1,J)=I$(J):NEXTJ:GOTO135
149 PRINT"▽CANCELLAZIONI S/N ":INPUTR$
151 IFR$<>"S"THEN163
153 REM
155 REM CANCELLAZIONE RECORD
157 SW=0:GOSUB375:IFSW<>0THEN163
159 SW=0:GOSUB387:IFSW<>0THEN155
```

```
161 MC$(K1,1)=SP$:GOTO155
163 REM
165 REM RISCRITTURA FILE
167 SW=0:FF=0:K1=0:N1=1
169 PRINT"▨PREPARA NUOVO NASTRO":GOSUB267
171 GOSUB275:GOSUB281
173 PRINT"▨INSERIMENTI S/N ":INPUTR$
175 IFR$<>"S"THEN201
177 REM
179 REM INSERIMENTO NUOVO RECORD
181 GOSUB213:IFSW<>0THEN199
183 IFFF=1THEN189
185 GOSUB233
187 IFU<>0THENU=0:GOTO193
189 IFI$(1)>LC$GOTO195
191 IFI$(1)=LC$ANDI$(2)>LN$THEN195
193 GOSUB317:GOSUB319:GOTO181
195 GOSUB221:LC$=I$(1):LN$=I$(2)
197 K1=K1+1:GOTO181
199 IFFF=1THEN203
201 GOSUB253
203 PRINT"▨FINITO AGGIORNAMENTO"
205 PRINT"SCRITTI ";K1;" RECORD":CLOSE1
207 PRINT"FILE: ";NF$
209 GOSUB319:GOSUB365:RUN
211 REM
213 REM LETTURA NUOVI DATI
215 SW=0:GOSUB325:IFSW=1THENRETURN
217 GOSUB337
219 RETURN
221 REM
223 REM SCRIVE NUOVO RECORD SU NASTRO
225 FORJ=1TONC
227 IFLEN(I$(J))=0THENI$(J)=" "
229 PRINT#1,I$(J);CH$;:NEXTJ:RETURN
231 REM
233 REM RICERCA POSIZIONE NUOVO RECORD
235 U=0:FORK=N1TON:IFMC$(K,1)=SP$THEN249
237 IFMC$(K,1)<I$(1)THEN247
239 IFMC$(K,1)=I$(1)ANDMC$(K,2)<I$(2)THEN247
241 IFMC$(K,1)=I$(1)ANDMC$(K,2)=I$(2)THEN245
243 N1=K:K=N:NEXTK:RETURN
```

```
245 U=1:PRINT"DATI UGUALI":GOSUB319:GOTO243
247 GOSUB303:K1=K1+1:LC$=MC$(K,1):LN$=MC$(K,2)
249 NEXTK
251 FF=1:RETURN
253 REM
255 REM TERMINA SCRITTURA FILE
257 IFN1=NANDFF=1THENRETURN
259 FORK=N1TON:IFMC$(K,1)=SP$THEN263
261 GOSUB305:K1=K1+1
263 NEXTK:RETURN
265 REM
267 REM RICHIESTA PREPARAZIONE NASTRO
269 PRINT"⌂MONTA NASTRO":GOSUB319
271 RETURN
273 REM
275 REM RICHIESTA NOME FILE
277 INPUT"⌂NOME FILE ";NF$:RETURN
279 REM
281 REM APERTURA FILE PER SCRIVERE
283 OPEN1,1,2,NF$:RETURN
285 REM
287 REM APERTURA FILE DI STAMPA
289 OPENNF,PR:RETURN
291 REM
293 REM APERTURA FILE PER LEGGERE
295 OPEN1,1,0,NF$:RETURN
297 REM
299 REM LETTURA RECORD DA NASTRO
301 FORJ=1TONC:INPUT#1,I$(J):NEXTJ:FS=ST:RETURN
303 REM
305 REM SCRITTURA RECORD NASTRO DA VETTORI
307 FORJ=1TONC
309 IFLEN(MC$(K,J))=0THENMC$(K,J)=" "
311 PRINT#1,MC$(K,J);CH$;:NEXTJ:RETURN
313 REM
315 REM MESSAGGIO FUORI ORDINE
317 PRINT"⌂FUORI ORDINE "I$(1)SP$I$(2):RETURN
319 REM
321 GETKEYA$:IFA$=""THEN321:REM ATTESA TASTO
323 RETURN
325 REM
327 REM INGRESSO NUOVI DATI
```

```
329 FORJ=1TONC:I$(J)=" ":NEXTJ
331 PRINT"▽";:J=1:GOSUB351
333 IFI$(1)="*"THENSW=1:RETURN
335 FORJ=2TONC:GOSUB351:NEXTJ:RETURN
337 REM
339 REM CONFERMA DATI E CORREZIONE
341 PRINT"▽CONFERMI S/N":INPUTR$
343 IFR$="S"THENRETURN
345 INPUT"QUALE CAMPO ";J
346 IFJ<1ORJ>NCTHEN345
347 PRINTD$(J)" ";:I$(J)=" ":INPUTI$(J)
349 GOSUB359:GOTO339
351 REM RICHIESTA CAMPO
353 PRINTJ;" ";D$(J);" ";:I$(J)=" "
355 INPUTI$(J):RETURN
357 REM
359 REM STAMPA DATI RECORD
361 PRINT"▽";:FORJ=1TONC
363 PRINTJ;" ";D$(J);" ";I$(J):NEXTJ:RETURN
365 REM
367 REM ULTIMO MESSAGGIO
369 PRINT"▽▽▽▽▽RIAVVOLGI IL NASTRO"
371 PRINT"SE NECESSARIO"
373 GOSUB319:RETURN
375 REM
377 REM RICHIESTA PRIMI DUE CAMPI
379 PRINT"RISPONDI * PER     USCIRE"
380 I$(1)="":I$(2)=""
381 PRINTD$(1);:INPUTI$(1)
383 IFI$(1)="*"THENSW=1:RETURN
385 PRINTD$(2);:INPUTI$(2):RETURN
387 REM
389 REM RICERCA RECORD
391 FORK=1TON
393 IFMC$(K,1)=I$(1)ANDMC$(K,2)=I$(2)THEN401
395 NEXTK:SW=1
397 PRINT"▽NON TROVATO ";I$(1);SP$;I$(2)
399 GOSUB319:RETURN
401 K1=K:K=N:NEXTK:RETURN
```

COMMENTO AL PROGRAMMA

Il programma e' organizzato in modo che dopo aver preso visione di un quadro video, devi premere un tasto per proseguire; la cosa e' realizzata con il sottoprogramma delle linee 319/323.

Linee1/5: commenti.

Linea 7: PR e' il "dn" della periferica di stampa, noi abbiamo usato 3 per far uscire sul video; puoi sostituire il dn della stampante se ne disponi. Nel caso tu abbia una stampante, dovresti migliorare il programma introducendo un contatore per il numero delle linee di stampa, in modo da cambiare foglio al momento giusto. Per leggere piu' comodamente sul video i dati, durante la stampa ti conviene tenere premuto il tasto CBM LOGO.

Linea 9: NF e' il numero logico dell file di stampa, lo "lfn"; noi abbiamo usato 4. In realta' per fare uscire i dati sul video potevamo evitare di aprirlo come file, lo abbiamo fatto per rendere piu' semplice il trasferimento dell'uscita su una stampante.

Linea 11: NC e' il numero dei campi del record; viene usato per definire le variabili con indice e per gestire i cicli.

Linee 13/15: dimensionamento del vettore di stringhe D$ che deve contenere le descrizioni dei campi del record, e del vettore di stringhe I$, che deve contenere i dati da scrivere nel record, quando questi vengono letti dalla tastiera.

Linea 17: CH$ contiene il carattere RETURN, da usare come separatore nella scrittura dei campi; usiamo sempre CHR$(13) per non avere problemi con la lunghezza del record. SP$ contiene 3 spazi.

Linee 19/25: le due frasi DATA contengono le descrizioni dei campi, che vengono trasferite nel vettore D$ con la linea 25. Se si aumenta il numero dei campi NC, si possono aggiungere altre linee DATA.

Linee 27/41: presentazione del menu' principale per scegliere la funzione da svolgere. La linea 39 accetta la risposta, la 41 la controlla e accetta solo: 1, 2, 3 e 9.
Linea 43: se R=9 (FINE) viene eseguita la subroutine che chiede di riavvolgere il nastro se necessario e il programma si ferma.
Linee 45/55: viene chiesto il numero totale di record da elaborare, N. Esso serve per predisporre la zona per memorizzare il file in caso di aggiornamento. Questo dato e' importante e quindi ne viene chiesta conferma.
Linee 57/59: viene dimensionata la matrice di stringhe MC$, di N righe e NC colonne per contenere il file, se si devono operare aggiornamenti.
Linea 61: scelta della funzione principale in base al valore di R.
Linee 63/89: fase di CREAZIONE ex novo del file; essa si esegue la prima volta. Alla 67 vengono eseguiti i sottoprogrammi per preparare il nastro, richiedere il nome del file e aprire il file in scrittura. Alla 69 vengono poste a stringa nulla le due variabili LC$ e LN$, che vengono usate per controllare la sequenza dei record caricati, in base ai primi due campi. Poi inizia ur ciclo, da percorrere al massimo N volte, per caricare i record richiedendo i dati dalla tastiera. Il sottoprogramma in 211 legge i nuovi dati dalla tastiera. Se ritorna con SW<>0, questo significa che il caricamento e' terminato, allora viene conservato in K1 il numero dei record scritti, che puo' anche essere inferiore ad N, e si va alla fine di questa fase. Se e' stato caricato un nuovo record, ne viene controllata la sequenza rispetto al precedente; se va bene il record viene scritto con il sottoprogramma in 221 e viene chiesto un nuovo record. Se la sequenza non va bene questo viene segnalato; il record non viene accettato e potra' essere aggiunto in una successiva fase di aggiornamento del file. In 87 e 89 viene segna-

lato quanti record sono stati caricati e in quale file e il programma viene rilanciato con RUN.
Linee 91/111: fase di STAMPA. Alla 95 viene aperto il file di stampa, richiesto di montare il nastro, richiesto il nome del file e aperto per leggere. Poi vengono letti i record uno alla volta con il sottoprogramma in 297, stampati ponendo sulla prima riga i primi due campi e sulla seconda gli altri quattro. Nel caso tu aumenti il numero dei campi, dovrai probabilmente modificare le istruzioni alle linee 103 e 105. Viene controllata la fine del file analizzando la variabile riservata di stato ST, trasferita al momento della lettura in FS, e, quando il file e' terminato, la fase si arresta e dopo i soliti messaggi viene rilanciato il programma con RUN.
Linee 113/129: inizia la fase di AGGIORNAMENTO. Come prima cosa viene trasferito in memoria, nella matrice MC$, il file, di cui e' stato chiesto il montaggio, il nome, ed e' stato aperto in lettura. Viene segnalato se il file e' piu' lungo del previsto numero di record, e in questo caso si ha uno STOP e ti consigliamo di ripartire dando per N un numero adeguato. Alla fine del caricamento viene segnalato il numero effettivo dei record letti. L'aggiornamento avviene in fasi successive; prima l'eventuale modifica dei record presenti in memoria, poi l'eventuale cancellazione di qualche record, e poi il caricamento dei nuovi record, durante la fase di riscrittura del file.
Linee 131/147: fase MODIFICA record. Se non ci sono modifiche passa alla fase seguente. Alla 139 chiede i primi due campi del record da modificare con il sottoprogramma in 375; se al ritorno SW<>0, significa che non ci sono piu' modifiche. Alla 141 ricerca in memoria il record voluto, se non lo trova SW<>0 e ne chiede un altro. Legge dalla memoria gli altri campi del record selezionato e propone sul video tutto il record chiedendo quali campi vuoi modificare.

Non devi modificare i primi due campi, perche' se lo fai alteri l'ordine dei record. Il programma riscrive in memoria solo i campi dal terzo in avanti, quindi anche se tu hai modificato i primi due, essi non vengono memorizzati. Per modificare i primi due campi devi cancellare il record e poi riscriverlo.
Linee 149/161: fase CANCELLAZIONE. Come nella fase precedente vengono chiesti i primi due campi per individuare il record ed esso viene ricercato. Se esso viene trovato, vengono scritti degli spazi al posto del primo campo; in fase di riscrittura, se il primo campo contiene spazi, il record non viene scritto.
Linee 163/171: viene predisposta la riscrittura del file su nastro, chiedendo di montare il nastro e quale nome registrare in apertura. Alla 167 vengono preparati i seguenti indicatori:
. SW=0, se diventa 1 significa che non ci sono piu' nuovi record da aggiungere,
. FF=0, se diventa 1 significa che sono stati gia' scritti tutti i record della matrice MC$,
. K1=0, contatore dei record che si scrivono,
. N1=1, puntatore alla posizione del record nella matrice MC$.
Linee 173/175: richiesta se ci sono inserimenti.
Linee 177/209: fase INSERIMENTO e RISCRITTURA. Alla 181 viene chiesto un nuovo record; se viene fornito, vengono scritti sul nuovo file tutti i record di MC$, che precedono nell'ordinamento il nuovo record, e poi viene scritto il nuovo record. Non sono accettati record doppi. Quando MC$ e' esaurito si procede solo con nuove acquisizioni; quando i nuovi record sono terminati si esaurisce MC$, se FF=0. La fase di riscrittura e' la piu' complicata di tutto il programma, dato che vogliamo mantenere l'ordinamento dei record. Alla fine viene segnalato quanti record si sono scritti e il nome del file.

Linee 211/219: sottoprogramma lettura nuovi dati da tastiera, servendosi dei sottoprogrammi in 325 e 337.
Linee 221/229: scrittura nuovo record su nastro; i dati sono quelli ricevuti da tastiera. Ti facciamo notare che se un dato stringa ha lunghezza zero, esso viene sostituito con una stringa di uno spazio. Questo e' necessario perche' sul nastro venga registrato un campo valido; la stringa nulla viene saltata in fase di lettura e produce uno scompenso nella struttura dei record, ognuno dei quali deve avere un numero prefissato di campi. Ricorda che se leggi da tastiera una stringa formata da uno o piu' spazi, essa viene conservata come stringa nulla; questo non succede se usi SHIFT-spazio. Se, invece scrivi A$=" ", la stringa A$ ha lunghezza 1 e quando la scrivi su nastro lo spazio viene registrato e conta come un campo.
Linee 231/251: ricerca posizione per il nuovo record in MC$. Viene segnalato se i dati sono fuori ordine o sono doppi. Se tutto bene vengono trasferiti sul file i record di MC$ che precedono quello nuovo.
Linee 253/263: viene portata su file l'ultima parte di MC$.
Linee 265/271: richiesta di preparazione nastro.
Linee 273/277: richiesta nome file.
Linee 279/283: apertura file per scrivere.
Linee 285/289: apertura file di stampa.
Linee 291/295: apertura file per leggere.
Linee 297/301: lettura record da nastro.
Linee 303/311: scrittura record prelevato da matrice MC$, con modifica eventuali stringhe nulle.
Linee 313/317: messaggio record fuori ordine.
Linee 319/323: attesa pressione tasto per proseguire.
Linee 325/335: ingresso nuovi dati da tastiera; accetta anche campi vuoti.
Linee 337/349: visualizzazione dati, richiesta conferma o correzione.

Linee 351/355: richiesta di un campo.
Linee 357/363: visualizzazione dati di un record.
Linee 365/373: messaggio finale di ogni fase.
Linee 375/385: richiesta dati primi due campi, pone SW=1 se rispondi con * al primo campo.
Linee 387/401: ricerca record in MC$.

Nel programma FILESEQ abbiamo usato la tecnica dei sottoprogrammi, cioe' abbiamo scritto sotto forma di sottoprogramma tutti i gruppi di istruzioni che devono essere eseguite piu' di una volta; questa tecnica, oltre che portare a un risparmio di memoria, evita errori di programmazione.

Riportiamo l'elenco delle variabili usate nel programma:

PR=3, dn periferica stampa
NF=4, lfn file di stampa
NC=6, numero campi del record
D$(NC), descrizioni campi record
I$(NC), dati del record
CH$=CHR$(13), carattere separatore RETURN
SP$, costante contenente 3 spazi
R$, stringa per risposte
R, numero della scelta
N, numero record
K, variabile controllo ciclo
MC$(N,NC), matrice per contenere i record in memoria
LC$, primo campo del record precedente
LN$, secondo campo del record precedente
SW, flag per controllare se e' stato fornito un nuovo record
K1, numero record caricati
NF$, nome del file
FS, per conservare la parola di stato ST
ST, parola di stato
J, variabile controllo cicli
FF, controllo per record MC$

N1, puntatore a MC$
U, controllo record da inserire uguali a record MC$

La gestione del programma e' molto semplice:
.vengono evidenziati messaggi di richiesta sul video e devi rispondere,
.quando il quadro video e' fisso, e la cassetta non e' attiva, per proseguire devi premere un tasto,
.per uscire dalle fasi di richiesta dati devi rispondere con * al primo dato,
.devi stare attento al cambiamento del nastro o al suo riavvolgimento per non perdere registrazioni.

Per adattare FILESEQ alle tue esigenze puoi apportare le seguenti modifiche:
. porre PR a un nuovo valore, per usare una stampante,
. modificare NC e modificare le linee DATA per le descrizioni dei campi.
Resta comunque un programma che mantiene i record in ordine in base ai primi due campi; questa e' una cosa piu' difficilmente modificabile, potresti pero' mantenere uno dei campi sempre uguale a SHIFT-spazio e quindi ordinare solo su uno.
Inoltre FILESEQ non accetta record doppi.
Puoi aggiungere al programma altre parti, per ottenere altre funzioni, e puoi fare uso di alcuni dei sottoprogrammi di uso generale gia' presenti.
Il programma FILESEQ lascia liberi 7807 byte per le variabili; devi fare un po' di conti per decidere il massimo numero di record gestibili con MC$.

11.8 RIEPILOGO

In questo capitolo ci siamo occupati di:

- .FINESTRE VIDEO
- .DEFINIZIONE FUNZIONI UTENTE
- .ESTRAZIONE NUMERI A CASO: RND
- .SCRITTURA E LETTURA MEMORIA CON POKE E PEEK
- .RICERCA ERRORI: TRON, TROFF, HELP, STOP
- .GESTIONE ERRORI CON TRAP
- .GESTIONE FILE DI DATI SU CASSETTA
- .ESEMPIO RIASSUNTIVO: PROGRAMMA GESTIONE ARCHIVIO SU CASSETTA

se qualche punto non ti risulta chiaro, ritorna ad esaminare precedenti capitoli e rivedi qualche programma della cassetta.

APPENDICE A

IL BASIC 3.5

A.1 INTRODUZIONE

Il BASIC e' un linguaggio per programmare un calcolatore elettronico di tipo:

.CONVERSAZIONALE,
.INTERPRETATIVO.

Il termine CONVERSAZIONALE significa che durante la stesura del programma, la sua prova e la sua esecuzione, l'utente puo' intervenire per controllare risultati intermedi, apportare modifiche, correggere errori segnalati dal sistema, e tutto questo e' ottenibile con facilita' e immediatezza.
Il termine INTERPRETATIVO significa che nella memoria del calcolatore sono costantemente presenti, oltre al programma scritto in BASIC:
.un programma di sistema, che si chiama INTERPRETE BASIC,
.un insieme di programmi di sistema, che costituiscono il SISTEMA OPERATIVO,
e che essi interagiscono tra loro mettendo in grado l'utente di lavorare.
Il SISTEMA OPERATIVO e l'INTERPRETE BASIC costituiscono l'interfaccia tra l'utente e il calcolatore, che ha un suo linguaggio, il linguaggio macchina, e un suo modo di funzionare, che possono essere ignorati dall'utente.

Nel COMMODORE 16 il SISTEMA OPERATIVO e l'INTERPRETE BASIC risiedono stabilmente in una zona di memoria ROM di 32K byte.

Il BASIC del COMMODORE 16 e' una implementazione (versione) molto potente di questo linguaggio; essa e' siglata 3.5.

L'alfabeto del BASIC e' formato dai seguenti caratteri:

.le 26 lettere dell'alfabeto

A	B	C	D	E	F	G	H	I	J	K	L	M
N	O	P	Q	R	S	T	U	V	W	X	Y	Z

.le 10 cifre decimali

0	1	2	3	4	5	6	7	8	9

.lo spazio

.19 caratteri speciali.

"	#	$	%	π	(	)	*	+	,
-	/	:	;	<	=	>	?	^	

tutti presenti sulla tastiera.

Tutti i caratteri disponibili da tastiera, anche quelli non citati sopra, possono essere usati all'interno delle stringhe.

Gli elementi del linguaggio sono:

.COMANDI,
.ISTRUZIONI,
.FUNZIONI,
.OPERATORI,
.VARIABILI,
.COSTANTI,

organizzati, secondo regole, in frasi o linee.

Le frasi del linguaggio possono essere scritte in due modi diversi:

.per ESECUZIONE IMMEDIATA,
.per ESECUZIONE DIFFERITA.

La distinzione viene fatta in base alla presenza di un numero all'inizio della linea; se non e' presente un numero vengono eseguiti subito, alla pressione del RETURN, i comandi e/o le istruzioni indicati. Se, invece, la linea inizia con un numero, essa viene memorizzata nell'area programma, inserendola al posto giusto in base al numero, per esecuzione differita.
Comunque la linea rappresenta un programma per il calcolatore, che puo' al limite consistere in un solo comando.
Qualora di una linea di programma facciano parte piu' comandi e/o istruzioni, essi devono essere separati dal carattere ":". La lunghezza di una linea di programma puo' essere di al massimo 88 caratteri.
I comandi, le istruzioni e le funzioni sono formate da PAROLE CHIAVE, che sono riservate, cioe' non possono essere usate per scopi diversi da quelli stabiliti, accompagnate, eventualmente, da parametri che ne precisano il significato.
Sono COMANDI quelle istruzioni che agiscono sulla preparazione e la verifica del programma, la sua memorizzazione su nastro o disco magnetico e il suo richiamo in memoria, la sua esecuzione, la situazione della memoria e la predisposizione dei tasti funzione. Sono ISTRUZIONI tutte le altre. Di norma i comandi vengono usati in modo immediato, ma possono essere usati anche in modo differito, quando la cosa abbia senso logico. Non tutte le istruzioni possono essere usate in modo immediato.
Le FUNZIONI sono istruzioni particolari che forniscono un risultato numerico o stringa, o di stampa.

Una linea di programma e' formata dai seguenti elementi:

.NUMERO DI LINEA, da 0 a 63999, manca per esecuzione immediata,

.PAROLE CHIAVE (comandi, istruzioni, funzioni),
.COSTANTI,
.VARIABILI,
.OPERATORI,
.CARATTERI SEPARATORI.

I caratteri separatori ammessi sono:
.lo spazio (che puo' essere anche omesso, con lo svantaggio di rendere difficile la lettura, ma con il vantaggio di fare risparmiare memoria),
.i due punti (":"),
.la virgola (","),
.il punto e virgola (";").

A.2 COSTANTI E VARIABILI

Le COSTANTI possono essere:
.NUMERI INTERI, che vengono considerati tali solo se compresi tra -32767 e +32767. I numeri interi non compresi nell'intervallo citato vengono considerati reali.
.NUMERI REALI, che vengono stampati in forma decimale fino a 9 cifre (in valore assoluto minori di o uguali a 999999999, e non compresi nell'intervallo -0.1/+0.1, escluso lo zero), in forma esponenziale oltre.
.STRINGHE di caratteri alfanumerici, che sono in generale delimitate dal carattere virgolette. Esse non devono superare i 255 caratteri, ma, se fanno parte di una linea di programma, devono avere una lunghezza che non provochi il superamento degli 88 caratteri consentiti per la linea. Nel seguito viene indicato quando una stringa di caratteri puo' essere scritta senza virgolette delimitatrici.

Le costanti possono essere introdotte direttamente nelle linee di programma, oppure possono essere assegnate a delle variabili e richiamate mediante il nome delle variabili.

Nella memoria del calcolatore sono contenute due costanti speciali; una e' richiamata dai tasti CBM-= ed e' π (π=3.14159265), l'altra si ottiene con la funzione EXP(1) ed e' il numero irrazionale "e", base dei logaritmi naturali (e=2.71828183); queste costanti sono memorizzate con valore approssimato.

Le VARIABILI possono essere dei seguenti tipi:
.NUMERICHE INTERE,
.NUMERICHE REALI
.STRINGA
.NUMERICHE INTERE CON INDICE,
.NUMERICHE REALI CON INDICE,
.STRINGA CON INDICE.
Il tipo delle variabili viene evidenziato dal nome; le regole per la formazione dei nomi sono:
.formati da lettere o cifre,
.il primo carattere deve essere una lettera
.possono essere lunghi a piacere,
.vengono distinti in base ai primi 2 caratteri,
.devono terminare con il suffisso "%" per indicare numeri interi,
.devono terminare con il suffisso "$" per indicare stringhe,
.la mancanza di suffisso indica variabili numeriche reali,
.non possono far parte del nome le parole chiave del BASIC,
.la presenza di una coppia di parentesi dopo il nome con all'interno dei numeri o delle variabili numeriche, separati da virgole, indica che si tratta di variabili con indice del tipo specificato dal nome. Gli indici sono considerati solo per la parte intera, con troncamento degli eventuali decimali; essi partono da 0, cioe' il primo elemento ha indice 0.

Quando viene lanciato un programma con il comando RUN le variabili numeriche sono inizializzate al valore 0 e le variabili stringa alla

stringa nulla. La stringa nulla e' una stringa che non contiene caratteri, di lunghezza 0; viene indicata da due virgolette vicine.
Le due stringhe A1$ e A2$ che seguono sono diverse:
A1$=" " contiene uno spazio e ha lunghezza 1,
A2$="" e' una stringa nulla di lunghezza 0.
Le variabili vengono riempite, cioe' contengono dati, per effetto di una delle seguenti operazioni:
.ASSEGNAZIONE, con o senza la parola chiave LET, A$="BELLO", N=56.34, P%=6547, LET B$="piove".
.LETTURA, realizzata o con l'istruzione READ dall'interno del programma, o con le istruzioni GET, GETKEY e INPUT da tastiera, o con le istruzioni GET# e INPUT# da una periferica esterna.

Le variabili, non con indice, contengono un singolo dato; esse possono essere chiamate VARIABILI SINGOLE. Le VARIABILI CON INDICE sono riferite a un gruppo di dati, esse hanno tutte lo stesso nome e vengono distinte tra loro in base agli indici; vengono chiamate anche ARRAY o MATRICI. Questa categoria di variabili risulta molto utile per trattare dati che appartengono a un insieme, che ha senso logico considerare globalmente. Il numero di indici da assegnare ad una variabile dipende dalle caratteristiche del gruppo di dati che la variabile deve rappresentare. Per una lista di dati basta un indice. Per dati raggruppati in una tabella bidimensionale, servono due indici, il primo rappresenta le righe, il secondo le colonne della tabella. Con 3 indici si possono trattare variabili visualizzabili in uno spazio a 3 dimensioni.
In questa implementazione del BASIC non esiste un limite teorico al numero di indici (dimensioni) assegnabili, in quanto questo puo' essere pari a 255, caso difficilmente verificabile. Inoltre, ogni dimensione puo' raggiungere il massimo degli interi positivi, cioe' 32767, co-

sa peraltro impossibile date le dimensioni della memoria del COMMODORE 16. Gli indici partono dal valore 0.
Esiste una istruzione per definire le variabili con indice, la DIM; essa deve essere usata per definire variabili per le quali almeno un indice superi il numero 10, cioe' gli 11 elementi. Per le variabili per le quali ogni indice raggiunge al massimo il valore 10, il BASIC provvede a una definizione implicita la prima volta che viene citata la variabile con indice.

Esempi di variabili:

BELLO numero reale, nome riconosciuto come BE
BESTIA numero reale, nome riconosciuto come BE
N% numero intero
B$ variabile stringa
C(23) numerica reale con indice, definita con DIM
B%(7) numerica intera con indice, puo' non essere definita con DIM (indice<11)
F(10,15,6) numerica reale con 3 indici, definita da DIM

Le VARIABILI BOOLEANE o LOGICHE sono variabili che nascono dall'esecuzione di particolari istruzioni nel corso del programma con il significato di VERO o FALSO; in realta', come vedremo, possono essere considerate di tipo aritmetico.

VARIABILI RISERVATE

Le variabili elencate nella Tabella B.4 sono riservate, cioe' non possono essere usate nei programmi per scopi diversi da quelli loro assegnati dal sistema. Vediamo il loro significato.

DS e' la variabile di stato per le operazioni disco e serve per leggere dal canale comandi.

Per ottenere un messaggio piu' leggibile, si puo' usare l'analoga variabile stringa DS$.

EL contiene il numero di linea dove si e' verificato un errore.
ER contiene il numero dell'ultimo errore che si e' verificato.
ERR$ e' il nome di una funzione che permette di vedere la descrizione dell'errore il cui numero e' in ER.

ST e' una variabile di stato che contiene dopo una operazione di INPUT o OUTPUT, non diretta alla tastiera o al video, un numero che con il suo valore da' notizia sull'esito dell'operazione stessa. Essa serve per riconoscere se l'operazione e' andata a buon fine, se si e' verificato un errore, o se si sono verificate particolari condizioni, come la fine di un file.

TI e TI$ si riferiscono all'orologio interno del COMMODORE 16.
TI viene azzerato al momento dell'accensione del calcolatore o del RESET; esso viene incrementato di 1 ogni 60-esimo di secondo, per cui con opportuni calcoli si puo' conoscere il tempo trascorso. TI puo' essere evidenziata con una istruzione di OUTPUT e il suo contenuto puo' essere trasferito in un'altra variabile, ma non puo' ricevere direttamente un valore.
TI$ e' una variabile stringa di 6 caratteri; essa contiene nei primi due caratteri le ore (da 0 a 23), nei successivi due i minuti (da 0 a 59) e negli ultimi due i secondi (da 0 a 59). Il suo contenuto dipende da TI, ma in essa si puo' scrivere per inizializzare l'orologio come si desidera. Quando si modifica TI$, viene automaticamente modificato TI. In conclusione TI$ puo' essere letta e scritta, TI puo' essere letta , ma scritta solo in modo indiretto tramite TI$. Quando TI$="235959" viene automaticamente azzerata TI e aggiornata TI$.

A.3 OPERATORI ARITMETICI, RELAZIONALI E LOGICI

Il BASIC consente di scrivere espressioni formate da COSTANTI e VARIABILI collegate tra loro da OPERATORI che possono essere di tipo ARITMETICO, RELAZIONALE, LOGICO.

Gli OPERATORI ARITMETICI sono:

+	somma (segno +)	-	sottrazione (segno -)
*	moltiplicazione	/	divisione
^	elevato a	(	aperta parentesi
)	chiusa parentesi		

Gli OPERATORI RELAZIONALI sono:

<	minore di	=	uguale a
>	maggiore di	<=	< di o = a
>=	> di o = a	<>	non uguale a

Gli OPERATORI LOGICI sono:

AND	e (l'uno e l'altro)
OR	o (l'uno o l'altro)
NOT	no (negazione)

Le espressioni vengono elaborate partendo da sinistra e andando verso destra, rispettando le regole di precedenza che seguono, riportate dalla piu' alta alla piu' bassa:

^	elevato a	Se nell'espressione
-	segno meno	figurano delle pa-
*/	moltipl. e div.	rentesi, esse sono
+-	somma e sottr.	elaborate dando la
>=<	relazione	precedenza a quelle
NOT	negaz. logica	piu' interne.
AND	AND logico	
OR	OR logico	

Nel calcolo delle espressioni numeriche ogni operazione tra due operandi da' come risultato

un numero reale; alla fine del calcolo il risultato finale viene assegnato alla variabile che sta a sinistra dell'uguale rispettandone il tipo.

L'unica operazione aritmetica eseguibile tra stringhe e' la somma; essa consiste nell'unione delle due stringhe operando.

Le espressioni numeriche possono avere come operandi anche funzioni numeriche; analogamente le espressioni che danno come risultato una stringa possono avere come operandi funzioni stringa.

Gli operatori relazionali vengono usati per confrontare tra loro due espressioni; questo confronto ha significato numerico per espressioni numeriche, mentre ha un significato basato sull'ordinamento per espressioni stringa. Nel confronto tra stringhe vengono confrontati tra loro i codici ASCII dei caratteri componenti partendo da sinistra. I codici sono tali da mantenere il normale ordinamento alfabetico.
Dal confronto tra due operandi qualunque nasce una VARIABILE LOGICA, che puo' avere il significato di VERO o di FALSO, ma che ha un corrispondente significato aritmetico, -1 per VERO e 0 per FALSO. Questo fatto consente di introdurre in espressioni numeriche anche espressioni di tipo relazionale.
Esempi:

```
N%=R1$>R2$  N%=-1 se R1$>R2$
            N%= 0 se R1$=R2$ oppure R1$<R2$
```

A$="PIPPO" B$="PIPPO ", risulta A$<B$, infatti B$ contiene uno spazio dopo la parola PIPPO.

B=12345 B%=12345, risultano uguali anche se B e' una variabile reale e B% una variabile intera.

I confronti tra risultati numerici di calcoli complessi possono presentare piccole differenze dovute allo svolgimento dei calcoli in binario nella memoria del calcolatore.

Gli OPERATORI LOGICI, detti anche BOOLEANI, possono collegare tra loro due espressioni di qualunque tipo. L'applicazione di questi operatori produce un risultato logico di VERO o di FALSO in una variabile che, dal punto di vista aritmetico, vale 0 per FALSO e risulta diverso da 0 (non sempre -1) per VERO.

Gli operatori logici effettuano i confronti bit a bit secondo le regole che seguono:

Oper.logica	Risult.aritm.	Risult.relazion.
1 AND 1	1	VERO
1 AND 0	0	FALSO
0 AND 1	0	FALSO
0 AND 0	0	FALSO
1 OR 1	1	VERO
1 OR 0	1	VERO
0 OR 1	1	VERO
0 OR 0	0	FALSO
NOT 1	0	FALSO
NOT 0	1	VERO

Tutti gli operatori logici lavorano su espressioni numeriche, di cui considerano la parte intera; per non avere risultati errati la parte intera deve essere compresa tra -32767 e +32767.

A.4 COMANDI, ISTRUZIONI, FUNZIONI

In questo paragrafo elenchiamo in ordine alfabetico i comandi, le istruzioni e le funzioni del

BASIC. Per ognuno di essi vengono indicate le parole chiave, gli eventuali parametri, il tipo (comando, istruzione, funzione), il modo (immediato o programma) nel quale puo' essere usato.

Nella spiegazione del linguaggio usiamo le seguenti convenzioni:

.Le PAROLE CHIAVE sono scritte in lettere maiuscole e non devono essere modificate (salvo le abbreviazioni riportate in Tabella B.2).

.I PARAMETRI (o ARGOMENTI) sono scritti in lettere minuscole.

.Le PARENTESI QUADRE delimitano argomenti opzionali.

.Le PARENTESI AD ANGOLO (simboli minore e maggiore) indicano che e' necessario scegliere uno degli argomenti presenti.

La BARRA INCLINATA separa argomenti tra i quali si puo' scegliere, senza introdurne altri.

I 3 PUNTINI indicano la possibilita' di ripetere piu' volte la sequenza.

Le VIRGOLETTE, se presenti devono essere mantenute.

Le PARENTESI ROTONDE se presenti devono essere mantenute.

Le abbreviazioni seguenti:

var	variabile,
esp	espressione,
numd	numero drive,
	numero disco nell'unita',
cond	condizione,

```
lfn        numero logico di un file,
nomef      nome di un file,
           abbreviato anche in fn,
unita'     numero logico di una periferica,
           abbreviato anche in dn.

Unita', numero logico della periferica vale:
  0 per la tastiera,
  1 per il registratore,
  3 per il video.
```

==========

ABS — Funzione numerica
Immediato/Programma

ABS(esp)
fornisce il valore assoluto di esp, che deve essere un'espressione numerica.

==========

ASC — Funzione numerica
Immediato/Programma

ASC(x$)
fornisce il codice ASCII del primo carattere della stringa contenuta nella variabile x$. L'argomento puo' essere una costante stringa tra virgolette. Il risultato e' un numero compreso tra 0 e 255. Si ha un messaggio di errore se l'argomento e' la stringa nulla; per evitarlo conviene aggiungere alla stringa argomento la stringa CHR$(0), cosi': PRINT ASC(x$+CHR$(0)).

==========

ATN — Funzione numerica
Immediato/Programma

ATN(esp)
fornisce l'angolo, espresso in radianti, che ha come tangente l'argomento, che deve essere un'espressione numerica. Per ottenere il risultato in gradi si deve procedere cosi': ATN(esp)*180/π.

==========

AUTO Comando Immediato

AUTO [incremento]
attiva la numerazione automatica delle linee di programma. Se manca incremento la numerazione automatica viene disattivata. L'incremento, che deve essere un numero intero, rappresenta la differenza tra due numeri di linea successivi. Il numero, da cui iniziare la numerazione automatica, dipende dal numero che viene attribuito alla linea di programma che si scrive subito dopo il comando AUTO. Quando si preme RETURN (cioe' viene accettata una linea di programma) compare automaticamente il numero della linea successiva.

AUTO 5 predispone l'incremento di 5 tra i numeri di linea.
AUTO annulla la predisposizione

Per fermare la numerazione basta rispondere con solo RETURN al numero di linea che compare, o con SHIFT-RETURN, se non si vuole cancellare la linea corrispondente.

==========

BACKUP Comando Immediato/Programma

BACKUP Dnumd1 TO Dnumd2 [,ON Uunita']
duplica un dischetto su un secondo dischetto; puo' essere usato solo con un sistema collegato a una doppia unita' disco, e non 2 unita' singole. Il comando formatta il dischetto destinazione, che perde il suo precedente contenuto. Si ottiene una copia integrale del dischetto montato su numd1 (disco sorgente) nel dischetto montato su numd2 (disco destinazione).

==========

BOX Istruzione
Immediato/Programma

BOX [colore],x1,y1[,x2,y2][,angolo][,pit]
disegna un rettangolo sul video, con le seguenti caratteristiche:

colore
: 0 o 1, default 1 (col. inchiostro) anche 2 e 3 in modo grafico multicolore

x1,y1
: cordinate di un angolo

x2,y2
: coordinate dell'angolo opposto

angolo
: rotazione in gradi (senso orario), default 0

pit
: pittura: 0 per OFF, 1 per ON (rettangolo colorato in colore); default 0

E' necessario porre delle virgole per segnalare gli argomenti intermedi omessi. Se e' presente solo una coppia di coordinate, l'angolo opposto e' la posizione del cursore grafico (pennino). Nel caso di una sola coppia di coordinate, dopo l'esecuzione dell'istruzione il cursore grafico non viene spostato dalla posizione precedente. Nel caso, invece, della presenza delle 2 coppie di coordinate il cursore grafico, dopo l'esecuzione dell'istruzione si trova nella posizione del secondo angolo specificato. L'istruzione da' risultati visibili solo se il calcolatore e' in uno dei modi grafici disponibili. Le coordinate possono essere influenzate dalla precedente esecuzione dell'istruzione SCALE.

==========

CHAR Istruzione
Immediato/Programma

CHAR[colore],x,y,"stringa"[,modo]
consente di scrivere una stringa in una deter-

minata posizione del video, le cui coordinate sono x,y, intese come posizione carattere (x da 0 a 39, y da 0 a 24). Puo' agire sia su un video predisposto per testo che per grafica. Solo che i caratteri di controllo eventualmente presenti nella stringa, gli stessi consentiti con l'istruzione PRINT, non agiscono nel video grafico. Nel video grafico+testo se Y>19 il COMMODORE 16 non mostra i caratteri richiesti, che pero' appaiono se si passa a un modo totalmente grafico.

colore
: 0 e 1, default 1 (col. inchiostro) anche 2 e 3 in modo grafico multicolore

x,y
: coordinate posizione iniziale

stringa
: testo da scrivere

modo
: 0 campo diretto, 1 campo inverso

==========

CHR$ — Funzione stringa — Immediato/Programma

CHR$(x)

fornisce una stringa di un carattere, il cui codice ASCII e' x, che deve essere compreso tra 0 e 255. Se l'argomento non e' intero, ne viene considerata solo la parte intera. La stringa ha lunghezza 1 e puo' anche essere non stampabile.

==========

CIRCLE — Istruzione — Immediato/Programma

CIRCLE[colore],[xc,yc],xr[,yr][,[ia][,[fa][,[angolo][,inc]]]]]

disegna un cerchio, un ellissi, un arco o un poligono in dipendenza dai parametri presenti.

colore

0 e 1, default 1 (col. inchiostro) anche 2 e 3 in modo grafico multicolore

xc,yc

coordinate del centro, default pos. del cursore grafico (pennino)

xr

lunghezza orizzontale del raggio

yr

lunghezza verticale del raggio, default xr, yr=xr per cerchi e poligoni regolari

ia

posizione partenza arco in gradi, default 0

fa

posizione finale arco in gradi, default 360

angolo

rotazione in gradi in senso orario, default 0

inc

ampiezza in gradi dell'angolo sotteso al lato del poligono regolare che approssima la circonferenza, default 2 gradi (180 lati)

L'istruzione agisce in modo visibile solo se e' stato selezionato un modo grafico. Se xr e yr sono diversi si ottengono ellissi. Per ottenere poligoni regolari si deve porre inc=(360/num.lati).

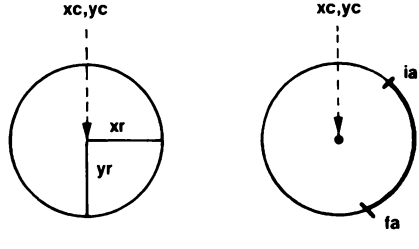


Figura A.1 Parametri CIRCLE

==========

CLOSE Istruzione
Immediato/Programma

CLOSE lfn
chiude un file, di numero logico lfn, aperto con OPEN, su una periferica. L'operazione CLOSE deve sempre essere eseguita a conclusione delle operazioni che riguardano file, pena il buon funzionamento del sistema e la perdita di dati.

==========

CLR Istruzione
Immediato/Programma

CLR
azzera tutte le variabili presenti in memoria, resettando i puntatori alle zone delle variabilied esegue una RESTORE, ma non azzera il video. Inoltre chiude tutti i file, ma non in modo corretto, pulisce l'area STACK, e, per quest'ultima ragione non deve essere usato all'interno di cicli o sottoprogrammi. L'istruzione CLR viene eseguita automaticamente dopo: NEW, RUN, o operazioni di EDIT (modifica anche apparente del programma presente in memoria).

==========

CMD Istruzione
Immediato/Programma

CMD lfn [,lista]
trasferisce l'uscita video su una periferica, sulla quale e' stato preventivamente aperto con OPEN il file logico lfn. Dopo CMD, PRINT e LIST agiscono sulla periferica selezionata, che puo' essere la stampante, il nastro o il disco.
Dopo l'esecuzione delle operazioni di OUTPUT, per chiudere correttamente la comunicazione e' necessario eseguire i comandi:
PRINT #lfn:CLOSE lfn.

==========

COLLECT — Comando Immediato/Programma

COLLECT [Dnumd] [,ON Uunita']
sistema un dischetto dove sono presenti file non correttamente chiusi e aggiorna la DIRECTORY e la BAM. Esegue una VALIDATE del dischetto.

==========

COLOR — Istruzione Immediato/Programma

COLOR sorgente, colore [,luminosita']
consente di assegnare un colore, mediante il suo numero a una delle 5 possibili sorgenti.
I numeri colore variano da 1 a 16, e sono quelli riportati sui relativi tasti numerici. Le possibili sorgenti sono numerate da 0 a 4, come segue:

Colore sfondo	Sorgente 0
Colore inchiostro	Sorgente 1
Modo multicolore 1	Sorgente 2
Modo multicolore 2	Sorgente 3
Colore bordo	Sorgente 4

La luminosita' puo' variare da 0 (bassa luminosita') a 7 (alta luminosita); il default e' 7. Non ha senso modificare la luminosita' del colore nero. Il richiamo del numero di una sorgente rende disponibile il colore che e' stato assegnato ad essa con COLOR. Alla partenza o dopo un RESET del calcolatore la situazione dei colori e' la seguente:

Sorg.0: col. 2, lum. 7	Sorg.1: col. 1, lum. 1
" 2: " 7 " 3	" 3: " 12, lum. 5
" 4: " 15 " 6	

==========

CONT — Comando Immediato

CONT
fa ripartire un programma che si e' arrestato per effetto delle istruzioni STOP o END, o per l'uso del tasto RUN/STOP. Se il programma si e' fermato per l'istruzione END non compare alcun messaggio. Negli altri due casi compare il messaggio BREAK IN num-linea. CONT puo' essere usato solo se, dopo l'arresto del programma, non si e' entrati in EDITOR. Per entrare in EDITOR basta portare il cursore su una linea di programma e premere RETURN, anche senza fare alcuna modifica. E', invece, possibile usare comandi in immediato per visualizzare variabili, modificare il valore di alcune variabili, chiedere il LIST di parte del programma, e poi usare il comando CONT per proseguire.

==========

COPY — Comando Immediato/Programma

COPY [Dnumd1,] "nomef1" TO [Dnumd2,] "nomef2"
[,ON Uunita']
copia il file di nome "nomef1" dal dischetto montato su numd1 nel file di nome "nomef2" sul dischetto montato su numd2. I dischetti possono essere diversi se e' collegata una doppia unita' disco, non 2 unita' singole. Se numd1=numd2 il comando produce una copia del file sullo stesso dischetto; in tale caso nomef1 deve essere diverso da nomef2.

COPY D0,"FILEA" TO D1,"FILEB"
copia il file "FILEA" dal dischetto montato sul drive 0 nel file "FILEB" sul dischetto montato sul drive 1.

COPY D0 TO D1 copia tutti i file dal dischetto montato sul drive 0 al dischetto montato sul drive 1

COPY "PRIMO" TO "SECONDO" copia sullo stesso dischetto il file "PRIMO" nel file "SECONDO".

==========

COS Funzione numerica
Immediato/Programma

COS (esp)
fornisce il coseno dell'argomento, che deve essere la misura di un angolo in radianti. Per trasformare i gradi in radianti basta moltiplicare per π/180.

==========

DATA Istruzione
Programma

DATA lista
consente di introdurre nel programma una serie di dati, che va a formare un file di dati interno al programma. La lista consiste in una serie di costanti di qualunque tipo, separate da virgola. Nel programma possono comparire diverse istruzioni DATA e possono essere posizionate dove si vuole. Quello che conta e' l'ordine con il quale le DATA si susseguono, infatti il file interno di dati viene formato secondo l'ordine delle frasi e l'ordine dei dati di ogni lista. Il programma al momento del RUN posiziona un PUNTATORE all'inizio del file di dati; esso viene spostato prima del dato successivo dopo la lettura di un dato con l'istruzione READ. Se si tenta di leggere piu' dati di quelli disponibili si ha segnalazione di errore. Esiste un'istruzione che consente di posizionare il puntatore all'inizio della lista dati di una particolare linea DATA; essa e' RESTORE.
Le costanti numeriche possono essere scritte in formato intero, con decimali e in formato

esponenziale; le stringhe devono comparire tra virgolette solo se contengono i seguenti caratteri:

,	:	spazi	SHIFT-lettera
caratteri grafici			caratteri di controllo

==========

DEC — Funzione numerica
Immediato/Programma

DEC(stringa)
fornisce il numero decimale corrispondente alla stringa, che deve essere una stringa, di al massimo 4 caratteri, rappresentante un numero esadecimale e puo' essere passata come una costante tra virgolette o come una variabile stringa.

```
PRINT DEC("FF")          stampa 255

A$="5F":PRINT DEC(A$)    stampa 95
```

==========

DEF FN — Istruzione
Programma

DEF FNnome(var)=esp
dove:
DEF FN puo' essere scritto anche DEFFN
nome
  e' il nome che si vuole assegnare alla funzione e deve essere di 2 caratteri, con il primo alfabetico
var
  e' il nome di una variabile reale che serve come argomento di comodo per la definizione matematica
esp
  e' un'espressione di calcolo che puo' comprendere anche altre funzioni, ma solo di tipo reale

La funzione deve essere definita nel programma prima di richiamarla. Dopo la definizione essa si comporta come le funzioni del BASIC. Per richiamarla si deve citare il nome: FNnome(x), dove x e' l'argomento vero per il quale si vuole eseguire il calcolo, e puo' essere una variabile numerica reale o una costante.

==========

DELETE — Comando Immediato

DELETE [num1] [-num2]
cancella le linee di programma BASIC dalla linea num1 alla linea num2.

DELETE 800 cancella la linea 800

DELETE 50-90 cancella dalla linea 50 alla linea 80

DELETE -100 cancella dall'inizio fino alla linea 100

DELETE 5000- cancella dalla linea 5000 fino alla fine del programma

==========

DIM — Istruzione Immediato/Programma

DIM var(ind) [,var(ind)...]
dove var e' uno dei nomi consentiti per le variabili e tra parentesi sono indicati i valori massimi di ogni indice, separati da virgole. Gli indici possono essere costanti o variabili numeriche, delle quali viene considerata la parte intera.
Deve essere usata DIM per definire variabili con indice per le quali almeno un indice superi 10. Gli indici partono dal valore 0, quindi l'indice 10 corrisponde a 11 elementi. Fino a 11

elementi il sistema fa una definizione implicita della variabile al primo richiamo. In un programma non e' consentito dimensionare due volte la stessa variabile. Teoricamente il numero massimo di dimensioni puo' essere 255 e il limite per ogni indice e' 32767; bisogna ovviamente scegliere valori che siano compatibili con la memoria disponibile e con la lunghezza della linea di programma.

==========

DIRECTORY Comando Immediato/Programma

DIRECTORY [Dnumd] [,Uunita'] [,"nomef"]
mostra la directory del dischetto montato sull'unita' collegata, senza distruggere il precedente contenuto della memoria. Puo' essere usato CTRL-S per fermare lo scrolling, che riprende alla pressione di un qualunque tasto. Per rallentare la visualizzazione si puo' premere il tasto CBM LOGO.
Per ottenere la copia su carta della DIRECTORY, se e' collegata una stampante, si deve procedere cosi:

```
LOAD"$0",8:OPEN4,4:CMD4:LIST
PRINT#4:CLOSE4
```

ma questa operazione distrugge il precedente contenuto della memoria.

DIRECTORY lista la directory dell'unico dischetto collegato

DIRECTORY D1,U9,"ANAGRAFICO" lista la linea relativa al file di nome ANAGRAFICO della directory del dischetto montato sul drive 1 dell'unita' 9.

==========

DLOAD Comando Immediato/Programma

DLOAD "nomef" [,Dnumd][,Uunita']

carica da disco il programma di nome "nomef". Puo' essere specificato il numero del drive, numd, necessario per unita' doppie; il numero dell'unita', che e' necessario nel caso di collegamenti multipli.
Invece del nome del programma, passato come costante stringa, puo' essere usata una variabile stringa inclusa tra parentesi.

```
N$="ANAGRAFICO":DLOAD(N$)

DLOAD"CONTABILITA'"
```

Se DLOAD e' usato da programma, dopo il caricamento il nuovo programma parte automaticamente, come per effetto di un GOTO alla prima linea. Devi fare attenzione perche' non viene riposizionato il puntatore all'inizio delle variabili (byte 45 e 46). Per poter lavorare correttamente con programmi concatenati, ogni programma, prima di caricare il successivo, deve eseguire un CLR e scrivere in 45 e 46 i valori adatti.
Tali valori si rilevano caricando per prova il programma da concatenare e leggendo con la funzione PEEK il contenuto dei byte 45 e 46.

==========

DO(LOOP) WHILE(UNTIL EXIT) Istruzione Immediato/Programma

DO [UNTIL cond/WHILE cond] Istruzioni[EXIT]
LOOP [UNTIL cond/WHILE cond]
esegue le istruzioni comprese tra DO e LOOP, fino a quando sono soddisfatte le condizioni.
Se mancano UNTIL e WHILE, sia dopo DO che dopo LOOP, le istruzioni vengono eseguite indefinitamente. Se viene trovato EXIT, il ciclo viene interrotto e l'esecuzione prosegue dall'istruzione successiva a LOOP.
La condizione cond e' un'espressione relazionale o una variabile booleana; la parola chiave

UNTIL significa "fino a quando la condizione diventa VERA, cioe' mentre e' FALSA", la parola chiave WHILE significa "fino a quando la condizione rimane VERA, termina appena diventa FALSA". Nella Figura A.2 sono riportati gli schemi logici.

Il fatto di poter porre la condizione da analizzare dopo DO o dopo LOOP, da' una grande liberta' nell'uso dell'istruzione. Ponendo la condizione subito dopo DO, il ciclo puo' non essere mai eseguito, mentre ponendola dopo LOOP, si ha almeno una esecuzione.

```
10 DO UNTIL X=5 OR X=7    esegue le istruzioni
15 INPUT "X=";X           da 15 a 25 fino alla
20 ...                    lettura di 5 o 7
25 ...
30 LOOP
```

```
DO WHILE A$="":GET A$:LOOP
```

legge un carattere da tastiera fino a quando viene premuto un tasto, cioe' prosegue dopo la pressione di un tasto

==========

DRAW — Istruzione Immediato/Programma

DRAW[colore][,x1,y1][,TO x2,y2][...]
consente di disegnare in modo visibile, se si e' in modo grafico, punti, linee e figure.

colore
: 0 e 1, default 1, (col. inchiostro) anche 2 e 3 in modo grafico multicolore

x1,y1
: coordinate del primo punto, se mancano il default e' la posizione attuale del cursore grafico (PENNINO)

x2,y2
: coordinate del secondo punto (finale) se manca TO x2,y2, viene disegnato solo un punto

Le coppie di punti iniziali e finali possono essere ripetute, ottenendo un disegno anche complicato. L'istruzione produce un effetto visibile solo se usata in modo grafico.

==========

DSAVE — Comando Immediato/Programma

DSAVE "nomef"[,Dnumd][,Uunita']
memorizza il programma presente in memoria sul disco, assegnandogli il nome "nomef". Per unita' doppie o collegamenti multipli devono essere specificati il numero del drive e quello dell'unita'. Puo' essere usata una variabile stringa al posto di "nomef", ma va inclusa tra parentesi.

DSAVE"PROVE" memorizza il programma con il nome PROVE

N$="ORDIN":DSAVE(N$) memorizza il programma con il nome ORDIN

==========

END — Istruzione Programma

END
interrompe l'esecuzione del programma senza evidenziare messaggi. Si puo' proseguire scrivendo, se e' possibile, CONT e premendo RETURN.

==========

EXP — Funzione numerica Immediato/Programma

EXP(esp)
fornisce il valore della costante e (base dei logaritmi naturali, numero irrazionale di cui e' memorizzata un'approssimazione = 2.71828183) elevata al numero che risulta dal calcolo di

esp. Se tale numero supera 88.0296919 si ha il messaggio di overflow.

==========

FOR...TO...STEP Istruzione Immediato/Programma

FOR var=val-i TO val-f [STEP inc]

dove:

var

variabile numerica reale, detta variabile di controllo del ciclo

val-i

valore iniziale per var

val-f

valore finale per var

inc

incremento per var, puo' essere positivo o negativo; default 1

val-i, val-f e inc possono essere anche espressioni numeriche semplici e non avere valore intero. Se val-f e' minore di val-i, inc deve essere negativo.

L'istruzione serve per INIZIARE L'ESECUZIONE di un CICLO; essa non puo' essere usata da sola; al termine del ciclo deve comparire l'istruzione NEXT var (puo' essere scritta anche senza var, e allora si riferisce all'ultimo FOR eseguito).

Il ciclo e' composto da tutte le istruzioni comprese tra FOR e NEXT.

Vediamo cosa avviene:

.1) con l'esecuzione di FOR viene inizializzato il contatore var con il valore val-i e viene memorizzato il valore finale val-f e l'incremento;

.2) vengono eseguite tutte le istruzioni comprese tra FOR e NEXT;

.3) al NEXT viene sommato al contatore var l'incremento inc, e il valore viene confrontato con val-f;

.4) se il valore raggiunto non supera val-f (o non risulta minore per incremento negativo), il

programma torna alla istruzione successiva al FOR (esegue un altro ciclo);
.5) se il valore raggiunto supera val-f (o e' inferiore) il programma prosegue con l'istruzione dopo il NEXT, cioe' esce dal ciclo.
All'uscita dal ciclo il valore di var e' superiore o inferiore a val-f. Inoltre il ciclo viene percorso almeno una volta, dato che la condizione viene analizzata al NEXT.

Se all'interno del ciclo sono presenti istruzioni condizionali (come IF) che fanno uscire dal ciclo, la variabile var deve essere forzata al valore finale e deve essere eseguito il NEXT, altrimenti il ciclo FOR resta pendente, non concluso, e questo puo' dar luogo a errori.

E' possibile avere cicli FOR uno interno all'altro, si dice nidificati, ma non si possono avere intrecci, ogni ciclo deve essere completamente interno al precedente. Con il COMMODORE 16 si possono nidificare fino a 10 FOR.

==========

FRE — Funzione di servizio
Immediato/Programma

FRE(x)
fornisce il numero dei byte liberi in memoria, x puo' essere qualunque, si tratta di un argomento di comodo.

==========

GET — Istruzione
Programma

GET lista variabili
legge un carattere per volta dalla tastiera.
Se non e' stato premuto alcun tasto legge la stringa nulla. E' bene che lista contenga nomi di variabili stringa (per evitare errori), separate da virgole se piu' di una.
GET accetta un tasto senza che debba essere

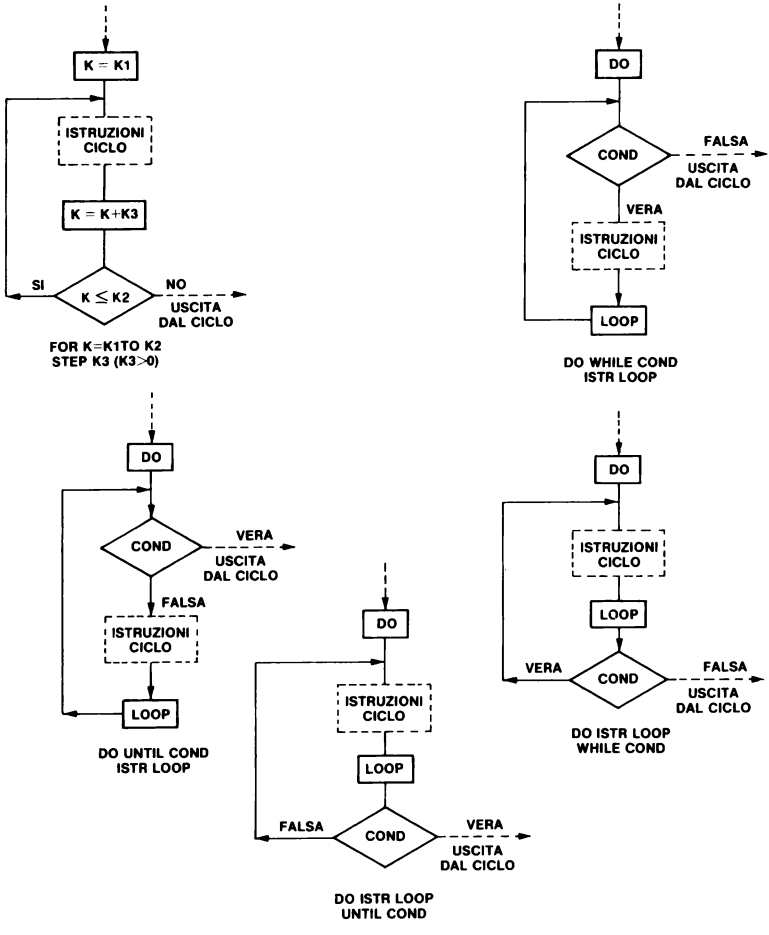


Figura A.2 Schemi logici FOR e DO...LOOP

seguito dalla pressione di RETURN.
L'istruzione puo' essere usata per creare cicli di attesa fino alla pressione di un tasto qualunque o di un tasto particolare:

10A$="":GETA$:IFA$=""THEN10 prosegue se si preme un tasto

10A$="":GETA$:IFA$=<>"P"THEN10 prosegue se si preme P

==========

GETKEY Istruzione Programma

GETKEY lista variabili
e' simile a GET solo che prosegue solo se si preme un tasto.

10GETKEYA$ prosegue se si preme un tasto

10GETKEYA$:IFA$<>"P"THEN10 prosegue se si preme P

Non devi premere uno dei tasti funzione in risposta a GETKEY, infatti da' luogo a segnalazione di errore.

==========

GET# Istruzione Programma

GET# lfn,lista variabili
dove:
lfn e' il numero logico di un file precedentemente aperto su una periferica
lista e' una lista di variabili separate da virgole
L'istruzione legge un carattere per variabile.
Con GET# si possono leggere tutti i caratteri, anche quelli di controllo.

==========

GOSUB Istruzione
Immediato/Programma

GOSUB num-linea
dove num-linea e' il nunero di una linea dove inizia un sottoprogramma.
Il controllo viene trasferito alla linea indicata, ma viene memorizzato il numero di linea e la posizione nella linea dell'istruzione GOSUB. Quando nel sottoprogramma viene incontrata l'istruzione RETURN, il controllo torna all'istruzione successiva a GOSUB.
All'interno di un sottoprogramma possono comparire altre istruzioni GOSUB; si dice che si ha il concatenamento dei sottoprogrammi. Si possono concatenare fino a 39 sottoprogrammi. Nella Figura A.3 e' riportato uno schema logico dell'istruzione.

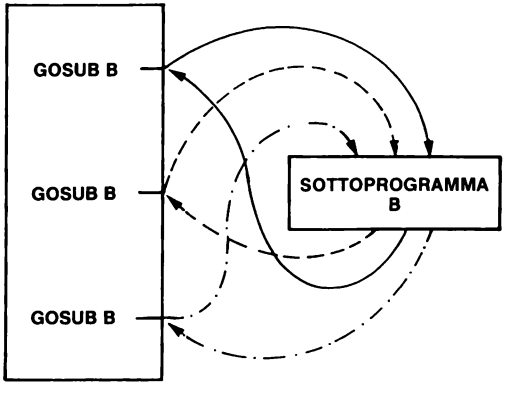


Figura A.3 Schema chiamata SOTTOPROGRAMMA

==========

GOTO — Istruzione Immediato/Programma

GOTO num-linea oppure GO TO num-linea
fa proseguire il programma da num-linea, che deve esistere nel programma, pena segnalazione di errore.

==========

GRAPHIC — Istruzione Immediato/Programma

GRAPHIC modo [,flag]
rende attivo per il video il modo grafico selezionato.
Flag puo' essere 1 per pulire il video, 0 per lasciarlo invariato.
I modi grafici sono:

0	testo
1	grafica ad alta risoluzione
2	grafica ad alta risoluzione e testo sulle 5 linee in basso
3	grafica multicolore
4	grafica multicolore e testo sulle 5 linee in basso

Quando si seleziona um modo da 1 a 4, il sistema assegna 10K byte alla grafica al disopra della zona dedicata al programma BASIC; in conseguenza restano poco piu' di 2K per il programma. Anche se dopo essere passati da un modo grafico si esegue GRAPHIC 0, i 10K restano allocati, per renderli disponibili di nuovo si deve usare l'istruzione GRAPHIC CLR.

==========

GSHAPE Istruzione Immediato/Programma

GSHAPE var-stringa [,[x,y][,modo]]

dove:

var-stringa

contiene l'immagine da mostrare

x,y

sono le coordinate dell'angolo in alto a sinistra da dove iniziare, default la posizione del cursore

modo

modo di visualizzazione:

0, senza modifiche (default)

1, campo inverso

2, con OR della zona

3, con AND della zona

4, con XOR della zona

==========

HEADER Comando Immediato/Programma

HEADER "nome-disco",Dnumd [,Iid][,ON Uunita']

e' l'operazione di formattazione dei dischetti. Essa deve essere eseguita sui dischetti nuovi o su quelli che si vogliono rendere usabili ex-novo. Con HEADER il dischetto oltre a ricevere un nome e una identificazione, viene suddiviso in tracce e settori e vengono registrati i relativi indirizzi, e inoltre viene predisposta una traccia per contenere l'indice dei contenuti (directory) e la tabella BAM di gestione dell'occupazione dei settori (blocchi). I parametri sono:

nome-disco

nome di al massimo 16 caratteri

numd

numero del drive su cui e' montato il dischetto

id
identificazione del dischetto, di 2 caratteri; e' necessaria per dischetti nuovi

unita'
numero dell'unita'

Se per un disco gia' in uso si omette Iid, si ottiene di cancellare il disco, ma vengono mantenuti gli indirizzi di traccia e settore e quindi l'operazione risulta piu' veloce.

==========

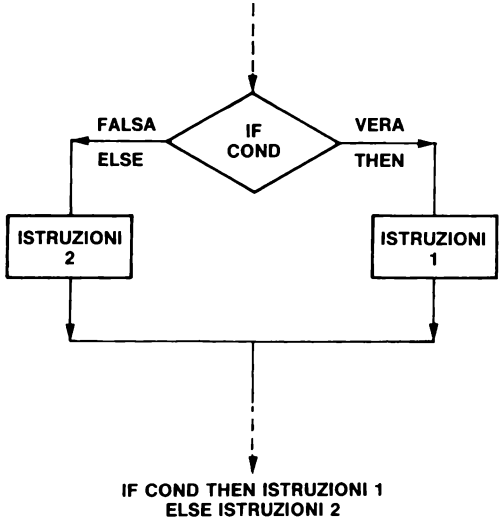
HEX$ — Funzione stringa
Immediato/Programma

HEX$(n)
fornisce una stringa di 4 caratteri che e' la rappresentazione esadecimale del numero n, che deve essere compreso tra 0 e 65535.

==========

IF...THEN...ELSE — Istruzione
Programma

IF cond THEN Istruzioni1 [:ELSE Istruzioni2]
dove cond e' una condizione.
Viene esaminata cond; se essa e' verificata, cioe' la condizione risulta VERA viene eseguito il gruppo di istruzioni che segue THEN (Istruzioni1) e al loro termine il programma prosegue dalla linea seguente. Se, invece, la condizione non e' verificata, cioe' risulta FALSA si possono avere due casi: se ELSE e' presente vengono eseguite le istruzioni dopo ELSE (Istruzioni2) e poi il programma prosegue dalla linea seguente, se ELSE non e' presente il programma prosegue dalla linea seguente.
La parte ELSE, se presente deve trovarsi sulla stessa linea di IF...THEN. Se i gruppi di istruzioni dopo THEN e/o dopo ELSE contengono delle istruzioni tipo GOTO, il programma non va alla linea successiva a IF.



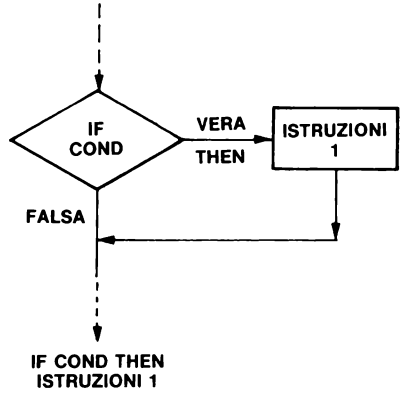


Figura A.4 Schema logico IF...THEN...ELSE

==========

INPUT Istruzione
Programma

INPUT ["messaggio";]lista variabili
e' l'istruzione per leggere dati dalla tastiera. Le variabili di lista devono essere separate tra loro da virgole. Il sistema fa apparire un punto interrogativo di richiesta dati; si deve rispondere con tanti dati quante sono le variabili, separandoli tra loro con virgole. Se il numero di dati e' inferiore al numero di variabili della lista il sistema richiede ancora dati con un doppio punto interrogativo. Se si risponde con dati di tipo che non concorda con il tipo delle variabili si ha un messaggio di errore e i dati vengono chiesti di nuovo.
L'immissione dei dati termina quando si preme RETURN. Se si risponde solo con RETURN, le variabili mantengono il loro precedente contenuto. Il messaggio tra virgolette, se presente, viene evidenziato prima del punto interrogativo di richiesta dati.
Se si risponde con piu' dati di quelli richiesti, quelli eccedenti vengono ignorati e il fatto viene segnalato.

==========

INPUT# Istruzione
Programma

INPUT# lfn, lista variabili
dove lfn e' il numero logico di un file precedentemente aperto su una periferica. L'istruzione e' simile alla precedente, solo che non puo' essere usato il messaggio, che d'altronde, non avrebbe senso.
L'istruzione preleva dati dal file considerando terminata una variabile quando incontra un separatore valido, che puo' essere la virgola (CHR$(44)) o il RETURN (CHR$(13)). Caratteri come "," e ":" all'interno di un dato stringa la

fanno considerare terminata; per questo motivo le stringhe contenti questi caratteri devono essere scritte forzando all'inizio e alla fine le virgolette con CHR$(34), cosi':
PRINT#lfn,CHR$(34);"stringa";CHR$(34)
I dati separati tra loro da virgole, invece che da RETURN, devono essere letti da una sola istruzione INPUT#, che puo' agire solo per i dati contenuti tra due RETURN, e questi non devono superare 88 caratteri; in caso contrario si perdono dati.

==========

INSTR — Funzione numerica
Immediato/Programma

INSTR(stringa1,stringa2[,pos-inizio])
ricerca in stringa1 la stringa2, partendo dall'inizio o da pos-inizio, se presente. Se non trova stringa2 da' risultato 0, se la trova da' la posizione di inizio di stringa2. Le due stringhe possono essere costanti o variabili.

==========

INT — Funzione numerica
Immediato/Programma

INT(x)
fornisce la parte intera di x troncando i decimali. Per arrotondare si deve operare cosi':
INT(x+0.5).

==========

JOY — Funzione numerica
Immediato/Programma

JOY(n)
fornisce la situazione di uno dei due joystick. Se n=1, del joystick 1, se n=2 del joystick 2. Il numero fornito risulta >128 se e' stato pre-

muto il bottone del fuoco e da' la direzione dello spostamento secondo lo schema seguente:

```
                alto
                  1
               8        2
   sinistra 7      0      3 destra
               6        4
                  5
                basso
```

alto+fuoco=129 destra+fuoco=131
numeri di posizione: 0, 1, 2, 3, 4, 5, 6, 7, 8.

==========

KEY Comando Immediato/Programma

KEY [num-k,stringa]
senza parametri fornisce l'elenco delle funzioni assegnate agli 8 tasti funzione (HELP corridsponde a F8). Con i parametri serve per assegnare una funzione a un tasto: num-k deve variare da 1 a 8 e individua il tasto, stringa deve essere la serie di funzioni da assegnare al tasto sotto forma di stringa.
Esempi:

```
KEY1,"OPEN4,4:CMD4:LIST"+CHR$(13)
KEY2,"PRINT#4:CLOSE4"+CHR$(13)
```

fanno si' che per listare un programma su stampante basta premere F1, e alla fine della stampa F2. Le assegnazioni iniziali sono:

```
KEY1,"GRAPHIC"
KEY2,"DLOAD"+CHR$(34)
KEY3,"DIRECTORY"+CHR$(13)
KEY4,"SCNCLR"+CHR$(13)
KEY5,"DSAVE"+CHR$(34)
KEY6,"RUN"+CHR$(13)
KEY7,"LIST"+CHR$(13)
KEY8,"HELP"+CHR$(13)
```

==========

LEFT$ Funzione stringa
Immediato/Programma

LEFT$(stringa, num)
dove:
stringa cost. o una var. stringa
num numero intero tra 0 e 255
fornisce i primi num caratteri della stringa.
Se num supera la lunghezza della stringa la fornisce tutta.

==========

LEN Funzione numerica
Immediato/Programma

LEN(stringa)
dove stringa puo' essere una costante o una variabile. Fornisce il numero dei caratteri della stringa, 0 per la stringa nulla.

==========

LET Istruzione
Immediato/Programma

[LET] var=esp
e' l'istruzione di assegnazione del BASIC. La parola chiave LET puo' mancare.
Il nome di var e il tipo dell'espressione devono concordare.

==========

LIST Comando
Immediato/Programma

LIST [numlinea1][-[numlinea2]]
lista sul video una parte o tutto il programma presente in memoria. La pressione del tasto CBM LOGO rallenta lo scrolling; per arrestare la lista basta premere CTRL-S, poi la pressione di qualunque tasto fa continuare.

LIST senza parametri lista tutto il programma

LIST numlinea una sola linea

LIST numlinea1- da numlinea1 in avanti

LIST -numlinea2 fino a numlinea2

LIST numlinea1-numlinea2
  il pezzo compreso tra i due numeri di linea

==========

LOAD Comando Immediato/Programma

LOAD ["nomef"[,unita'][,flag]]
carica un programma da nastro o da disco. Dove:
nomef
  nome del programma e puo' anche essere una variabile stringa
unita'
  numero logico dell'unita':
  1 per la cassetta e 8 per il disco
flag
  flag di rilocazione, vale 0 per rilocare (default), 1 per non rilocare (si veda comando SAVE)

LOAD carica il primo programma presente su nastro

LOAD "GIOCO",1 carica da cassetta il programma GIOCO

LOAD "PRIMO",8 carica da disco il programma PRIMO

Quando si usa da programma questo comando, dopo il caricamento il nuovo programma parte automaticamente, senza eseguire il CLR, quindi corrisponde a LOAD + GOTO prima linea.
Vedi il comando DLOAD per il problema della sistemazione dei byte 45 e 46.

==========

LOCATE — Istruzione Immediato/Programma

LOCATE x,y
consente di posizionare il cursore grafico (pennino) in un punto di coordinate x,y; se ne puo' utilizzare l'effetto solo con video in modo grafico.

==========

LOG — Funzione numerica Immediato/Programma

LOG(esp)
calcola il logaritmo in base "e" dell'argomento, che deve essere un numero non nullo e positivo.

==========

MID$ — Funzione stringa Immediato/Programma

MID$(a$,n1,n2)
puo' essere usata in due modi:
.1) per estrarre una parte di una stringa, se si trova a destra del carattere "=" in un'istruzione di assegnazione o in una lista di output,
.2) per modificare una parte di una stringa, se si trova a sinistra del carattere "=" in un'istruzione di assegnazione (pseudo variabile).
I parametri sono:

a$
: stringa su cui operare, nel caso 1) puo' anche essere una espressione stringa o una costante

n1
: cost. o var. numerica la cui parte intera da' la posizione da cui iniziare a operare

n2

cost. o var. numerica la cui parte intera da' il numero dei caratteri su cui operare

Il parametro n2 puo' essere omesso; allora nel caso 1) la funzione ritorna la parte destra della stringa a partire dal carattere di posizione n1. Nel caso 2) la presenza di n2 risulta inutile in quanto vengono sempre trattati i caratteri corrispondenti alla stringa sostitutiva.

```
100 A$="BELLA  GIORNATA":PRINT A$
105 MID$(A$,1)="BRUTTA":PRINT A$
```

sostituisce i caratteri da 1 a 6 della stringa A$ con "BRUTTA".

```
100 A$="OGGI PIOVE"
105 PRINT MID$(A$,6,5)
```

estrae "PIOVE" da A$ e lo stampa.

```
100 DATA BELLISSIMO,BRUTTISSIMO,INDIFFERENTI
105 FOR K=1 TO 3
110 READ A$:PRINT MID$(A$,1,5);" ";
115 NEXT K:PRINT
```

da' come risultato:

```
BELLI BRUTT INDIF
```

Nel caso di riassegnazione di parte di una stringa non deve venire modificata la lunghezza originale.

==========

MONITOR Istruzione Immediato/Programma

MONITOR

consente di uscire dal BASIC e rende disponibile il programma MONITOR. Il MONITOR permette di lavorare in linguaggio macchina, cioe' e' possibile accedere alla memoria, disassemblare parti dell'INTERPRETE BASIC e del SISTEMA

OPERATIVO, scrivere programmi in assembler, disassemblare programmi in linguaggio macchina, trasferire parti di memoria. Per tornare al BASIC pasta scrivere X e premere RETURN.

==========

NEW — Comando Immediato/Programma

NEW
serve per cancellare il programma BASIC presente in memoria e tutte le sue variabili. Di norma si usa prima di cominciare a scrivere un nuovo programma. Se viene usato da programma, questo viene interrotto e cancellato.

==========

NEXT — Istruzione Immediato/Programma

NEXT [var,...,var]
deve essere usato dopo FOR...TO, si veda FOR.
Si puo' scrivere senza var, con solo una var, o con piu' var separate da virgole.

==========

ON...GOTO (GOSUB) — Istruzione Programma

ON esp <GOTO/GOSUB> n1[,n2,...]
dove:
esp
e' un espressione numerica che deve dare un risultato non negativo, di cui viene considerata solo la parte intera
ni
sono i numeri di linea a cui trasferire il controllo
Se esp vale 1 il controllo e' trasferito a n1, se vale 2 a n2, e cosi' via. Se il valore di esp supera il numero dei numeri di linea presenti

oppure e' 0, il programma prosegue dalla linea seguente ON. Se, invece, il valore di esp e' negativo si ha un messaggio di errore.

==========

OPEN — Istruzione Immediato/Programma

OPEN lfn [,unita'[,sa[,"nomef,tipo,modo"]]]

dove:

lfn

numero logico file, da 1 a 255

unita'

numero della periferica:

0 per tastiera 1 per cassetta
3 per video 4 per stampante
8 per disco

sa

indirizzo secondario che ha significato diverso a seconda delle periferiche; per la cassetta: 0 per leggere, 1 per scrivere, 2 per scrivere con segnalazione di fine-nastro, per il disco rappresenta il numero del canale, da 2 a 14, 15 per i comandi, per la stampante rappresenta il modo di stampa

nomef

nome del file di al massimo 16 caratteri

tipo

serve solo per i dischi e puo' essere: PRG, SEQ, REL, USR

modo

serve solo per i dischi e puo' essere: READ o WRITE.

OPEN 1,3 apre il video come Input o come Output

OPEN 2,0 apre la tastiera come Input

OPEN 1,1,0,"PIPPO" apre la cassetta per leggere il file PIPPO

OPEN15,8,15 apre su disco il canale comandi

OPEN2,8,2,"ANAGRAFICO,SEQ,WRITE" crea un file sequenziale su disco

==========

PAINT — Istruzione Immediato/Programma

PAINT [colore] [,[x,y] [,modo]]

dove:

colore

0 o 1, default 1 (col. inchiostro) anche 2 e 3 in modo grafico multicolore

x,y

coordinate punto iniziale, default posizione cursore grafico (pennino)

modo

0 per bordo del colore=1

1 per bordo di un colore diverso dal colore dello sfondo

Serve per colorare parti di video delimitate da un bordo; il riempimento con il colore si arresta quando viene incontrato il bordo. Se il punto di partenza e' gia' del colore selezionato, non si ha pittura. Dopo l'esecuzione dell'istruzione il cursore grafico ritorna nel punto da cui e' partito per colorare. Produce effetti immediatamente visibili solo se usato in uno dei modi grafici. Le coordinate possono essere influenzate dalla precedente esecuzione di SCALE.

==========

PEEK — Funzione numerica Immediato/Programma

PEEK(n)

fornisce il contenuto del byte di indirizzo n, che puo' essere una costante o una variabile numerica, di cui interessa la parte intera, che deve essere compresa tra 0 e 65535.

==========

POKE — Istruzione
Immediato/Programma

POKE n,x
scrive nel byte di indirizzo n il numero x. I limiti per n sono da 0 a 65535 e per x da 0 a 255. Sia n che x possono essere costanti o variabili numeriche di cui interessa la parte intera.

==========

POS — Funzione numerica
Immediato/Programma

POS(x)
dove x e' un argomento qualunque. Fornisce la posizione di colonna del cursore di testo, da 0 a 39.

==========

PRINT — Istruzione
Immediato/Programma

PRINT [lista]
dove lista e' una sequenza di elementi di stampa, che possono essere: costanti, variabili, funzioni, caratteri di controllo, separati tra loro da caratteri separatori validi. I caratteri separatori possono essere:
. Virgola, che ha l'effetto di mandare dopo la stampa alla prossima zona di stampa. Sul video le zone di stampa sono 4 per linea, ognuna di 10 caratteri.
. Punto e virgola, che non aggiunge spazi dopo la stampa.
. Spazio, che viene ignorato e non provoca aggiunta di spazi, ma puo' generare ambiguita' di interpretazione; per esempio: A B viene interpretato come una variabile, la AB e non come due variabili.

Se la lista di stampa termina senza punteggiatura (, o ;) si ottiene di andare a nuova linea dopo la stampa, cioe' l'assenza di punteggiatura corrisponde alla sequenza: CHR$(13); (cioe' RETURN, ma seguito dal punto e virgola).
Ogni elemento viene stampato con il suo formato, che per le stringhe e' il loro numero di caratteri, mentre per i numeri viene aggiunto uno spazio prima per il segno + o il segno meno, e uno spazio dopo. Una PRINT senza lista dati manda a nuova linea.
Questa istruzione e' di norma diretta al video; l'uso precedente di OPEN e di CMD puo' trasferire l'uscita a un'altra periferica; non viene in generale rispettata in questo caso la dimensione delle zone di stampa.
Le funzioni che influenzano il posizionamento per la stampa sono TAB, SPC.

==========

PRINT# Istruzione Immediato/Programma

PRINT# lfn,lista
dove lfn e' il numero logico di un file precedentemente aperto e lista e' una lista di elementi da scrivere. Per ogni periferica si possono avere differenze di comportamento riguardo ai codici di controllo e alle funzioni di posizionamento e ai caratteri separatori. E' importante rilevare che quando l'uscita e' su supporti magnetici, come nastri e dischi, tra gli elementi della lista devono comparire dei caratteri separatori, passati come stringa, in modo che vengano registrati e consentano di riconoscere, in fase di lettura, la fine delle variabili. I dati sono inviati in uscita carattere per carattere, l'incontro di un carattere separatore valido fa ritenere terminato il dato in fase di lettura. I caratteri separatori riconosciuti sono la virgola (CHR$(44)) e il

RETURN (CHR$(13)) per tutti i tipi di dati; pero', se all'interno di una stringa si trova il carattere due punti o virgola, la stringa e' considerata terminata, per evitare cio', stringhe di questa tipo devono essere registrate con il primo e l'ultimo carattere forzati a virgolette, con CHR$(34).

==========

USING Istruzione Immediato/Programma

USING formato;lista
viene usata in congiunzione con PRINT o PRINT# (PRINT USING...) e serve per modificare il formato dei dati in uscita.
La lista e' la solita lista di dati da stampare: il formato e' una stringa di caratteri di controllo che svolgono specifiche azioni durante la stampa.
Il formato viene utilizzato per i dati della lista, eventualmente usandolo ripetutamente.
Segue l'elenco dei caratteri di controllo disponibili per i dati numerici:

predispone lo spazio per un carattere, se il dato eccede il numero di #, l'intero dato viene sostituito da tanti * quanti sono i #.

+ - possono essere posti nella prima o nell'ultima posizione di un campo. Il + fa stampare il segno + o il segno -. Il - fa stampare solo il - per i numeri negativi, per i positivi appare uno spazio. Se non si usa o il + o il -, il numero negativo viene preceduto dal - occupando una delle posizioni previste per le cifre.

. definisce la posizione del punto decimale; se non e' stato previsto il punto decimale, il numero viene arrotondato a intero.

, consente di far apparire una virgola nella stessa posizione del numero; si possono predisporre piu' virgole e almeno un carattere # deve precedere la prima virgola. Nella stampa vengono evidenziate solo le virgole che hanno una cifra a sinistra, le altre sono ignorate. Se il numero e' negativo viene stampato il segno meno.

$ fa stampare il carattere $ dove indicato; se si vuole che il carattere $ si sposti e preceda la prima cifra significativa del numero, esso deve essere preceduto da un #. Se si usa anche + o -, il segno precede il carattere $ in stampa.

^^^^ fanno si che il numero venga stampato in formato esponenziale. Se si usa anche # per definire il numero di caratteri, queste 4 frecce devono comparire dopo #. Per i numeri in formato esponenziale con caratteristica negativa si devono usare sempre ^^^^.

I caratteri di controllo per i campi stringa sono:

il numero di # determina il numero di caratteri della stringa da evidenziare, con eventuale troncamento a destra.

= serve per evidenziare una stringa al centro di un campo. Le dimensioni del campo sono date dal numero di # piu' 1, il carattere = puo' comparire ovunque.

> consente di evidenziare la stringa allineata a destra nel campo. I caratteri # definiscono le dimensioni del campo, con eventuale troncamento a sinistra.

==========

PUDEF — Istruzione
Immediato/Programma

PUDEF "stringa"

consente di modificare 4 caratteri di controllo di USING; i seguenti: spazio, virgola, punto decimale e dollaro. Per rimpiazzare questi 4 caratteri con altri quattro si deve preparare una stringa che rechi nella prima posizione il carattere sostitutivo dello spazio, nella seconda quello della virgola, ecc.

PUDEF " .,£" consente di stampare i numeri nell'abituale modo italiano, con la virgola prima dei decimali, il punto a separare le migliaia e il carattere £ al posto del $.

==========

RCLR — Funzione numerica
Immediato/Programma

RCLR(n)
dove n e' il numero di una sorgente di colore, da 0 a 4. Fornisce il colore disponibile nella sorgente specificata.

==========

RDOT — Funzione numerica
Immediato/Programma

RDOT(n)
dove n vale 0 per coordinata x, 1 per coordinata y e 2 per sorgente colore. Fornisce informazioni sulla posizione del cursore grafico. Ha senso usarlo se e' stato prima reso attivo un modo grafico.

==========

READ Istruzione
Programma

READ lista
consente di leggere i dati memorizzati nel file interno al programma con le istruzioni DATA. La lista e' formata da nomi di variabili separate da virgole. E' necessario che il tipo delle costanti concordi con il tipo delle variabili che vengono via via lette; altrimenti si ha segnalazione di errore. Si ha segnalazione di errore anche se si tenta di leggere piu' dati di quelli disponibili. Per ogni dato letto il puntatore interno nel file avanza di un dato.

==========

REM Istruzione
Programma

REM messaggio
consente di introdurre commenti in un programma. Dopo REM il messaggio puo' contenere qualunque carattere. Questa istruzione deve essere o l'unica o l'ultima di una linea di programma.

==========

RENAME Comando
Immediato/Programma

RENAME [Dnumd,] "nomev" TO "nomen" [,Uunita']
consente di cambiare nome a un file su disco; "nomev" viene sostituito con "nomen" nella DIRECTORY del dischetto.

==========

RENUMBER Comando
Immediato

RENUMBER [numeron [,inc [,numerov]]]
consente di rinumerare le linee di un programma partendo da numerov, che diventa numeron, e

usando l'incremento inc. Se non si usano parametri la rinumerazione inizia con 10 e l'incremento e' 10. Vengono aggiornati tutti i richiami a numeri di linea presenti nel programma.
La struttura dell'istruzione consente rinumerazioni totali o parziali.

==========

RESTORE Istruzione Programma

RESTORE [n]
consente di riposizionare il puntatore interno ai dati memorizzati con le istruzioni DATA. Il riposizionamento avviene alla prima linea DATA, se non compare n, alla linea n, se presente questo parametro.

==========

RESUME Istruzione Programma

RESUME [n/NEXT]
consente di ritornare al programma dopo una routine di errore. Se si usa senza parametri il ritorno e' all'istruzione che ha causato l'errore. Con il parametro puo' essere alla linea n, oppure, con NEXT, all'istruzione successiva a quella che ha causato l'errore.

==========

RETURN Istruzione Programma

RETURN
deve essere presente alla fine logica (conclusione) di un sottoprogramma; quando viene eseguito, il controllo del programma ritorna alla istruzione seguente la GOSUB. Se si incontra RETURN, senza aver eseguito prima un GOSUB, si ha segnalazione di errore.

==========

RGR Funzione numerica
Immediato/Programma

RGR(x)
dove x puo' essere qualunque, fornisce il numero del modo grafico corrente.

==========

RIGHT$ Funzione stringa
Immediato/Programma

RIGHT$(a$,x)
fornisce gli x caratteri piu' a destra della stringa a$. Il numero x puo' variare tra 0 e 255; a$ puo' essere una qualunque espressione stringa.

==========

RLUM Funzione numerica
Immediato/Programma

RLUM(n)
dove n e' il numero di una sorgente di colore, fornisce il grado di luminosita' della medesima, come numero.

==========

RND Funzione numerica
Immediato/Programma

RND(x)
fornisce un numero pseudo-casuale, compreso tra 0 e 1, prelevandolo dalla sequenza generata dal calcolatore mediante un algoritmo pseudo-casuale. La sequenza viene generata partendo da un numero iniziale chiamato "seme".
L'argomento x negativo, fa si che il seme venga posto sempre allo stesso valore (per lo stesso x), e quindi fa generare gli stessi numeri ogni volta che si usa il programma.
L'argomento x=0 fa si che il seme venga prele-

vato da TI, e quindi dipende dal tempo trascorso dall'accensione del calcolatore.
L'argomento x positivo fa si che la sequenza prosegua e quindi i numeri dipendono dalla precedente inizializzazione.

==========

RUN — Comando Immediato/Programma

RUN [n]
fa partire un programma, dall'inizio se manca n, dalla linea n se presente il parametro. Prima della partenza del programma viene eseguito un CLR.

==========

SAVE — Comando Immediato/Programma

SAVE["nomef"][,unita'][,flag]
memorizza il programma presente in memoria su nastro o su disco. I parametri sono:

nomef
: nome da assegnare al programma, puo' essere anche una variabile stringa

unita'
: numero logico periferica, 1 per la cassetta, 8 per il disco

flag
: 0 o assenza del flag, per memorizzare in modo rilocabile al momento del LOAD
: 1 per memorizzare mantenendo invariato il punto di caricamento
: 2 per il nastro per aggiungere EOT (fine nastro)
: 3 per il nastro per combinare gli effetti 1 e 2

Per il nastro puo' mancare il nome, ma non e' consigliabile memorizzare senza un nome.

Se il flag viene omesso esso vale 0 e ha il

significato di memorizzare il programma in modo rilocabile. La RILOCAZIONE di un programma ha questo significato:
. di norma il programma BASIC inizia al byte di indirizzzo 4097 e questo indirizzo si trova nei byte 43 e 44 (puntatore all'inizio del programma),
. se prima di scrivere un programma sposti il puntatore all'inizio del programma modificando il contenuto dei byte 43 e 44, il programma inizia a un indirizzo diverso da 4097,
. se usi il flag=0 nell'istruzione SAVE il programma viene memorizzato in modo rilocabile,
. se usi il flag=1 o il flag=3 nell'istruzione SAVE il programma viene memorizzato mantenendo l'informazione sul suo indirizzo di inizio, qualunque esso sia, e non e' rilocabile,
. al momento del LOAD, istruzione nella quale e' possibile usare per il flag il valore 1, o il valore 0 (assenza di flag), si ha il seguente comportamento:
.. per flag=0 il programma viene caricato a partire dall'indirizzo di inizio programma che si trova nei byte 43 e 44, rilocandolo o meno a seconda del modo come era stato memorizzato
.. per flag=1 il programma viene caricato a partire dall'indirizzo dove si trovava al momento del SAVE, e questo puo' essere in disaccordo con il valore contenuto nei byte 43 e 44.
Se per memorizzare e' stato usate il flag, l'istruzione VERIFY deve essere scritta in modo tale che non ci sia contrasto con la situazione della memoria e quello che sta sul nastro.

SAVE "PROVE" memorizza su nastro con nome PROVE

SAVE memorizza su nastro senza nome

SAVE "PROVA",8 memorizza su disco con nome PROVA

A$="MOVI":SAVE A$,8 memorizza su disco con nome MOVI

SAVE "ANALISI",1,2 memorizza su nastro con EOT final

==========

SCALE Istruzione
Immediato/Programma

SCALE <1/0>
ha un effetto immediatamente visibile solo se usata in uno dei modi grafici; permette di introdurre un fattore di scala per le coordinate.
Di norma, o per effetto di SCALE 0, i limiti per le coordinate grafiche sono:
in modo alta risoluzione 0<=x<=319 e 0<=y<=199
in modo multicolore 0<=x<=159 e 0<=y<=199

dopo l'esecuzione di SCALE 1 le coordinate x e y possono variare da 0 a 1023. Questo significa che vengono accettati per x e y valori nel range 0-1023 e pensa il sistema ad aggiustare i valori per far entrare i punti nel quadro video.

==========

SCNCLR Istruzione
Immediato/Programma

SCNCLR
pulisce lo schermo anche in modo grafico.

==========

SCRATCH Comando
Immediato/Programma

SCRATCH "nomef"[,Dnumd][,Uunita']
cancella il file di nome "nomef" dalla directory del dischetto montato sul drive numd della periferica di numero logico unita'. Se mancano gli ultimi 2 parametri deve essere collegata una

unita' a disco singolo. Per evitare cancellazioni involontarie il sistema chiede conferma con il messaggio "ARE YOU SURE?"; rispondendo Y avviene la cancellazione, rispondendo N no.

==========

SGN — Funzione numerica
Immediato/Programma

SGN(esp)
dove esp deve essere un'espressione numerica. Fornisce il segno dell'espressione:
0 se esp vale 0, 1 se e' positiva, -1 se e' negativa.

==========

SIN — Funzione numerica
Immediato/Programma

SIN(x)
fornisce il seno di x, che e' espresso in radianti.

==========

SOUND — Istruzione
Immediato/Programma

SOUND voce, frequenza, durata
produce un suono usando una delle due voci disponibili, con una frequenza che puo' variare da 0 a 1023 e con una durata, in 60-esimi di secondo, che puo' variare da 0 a 65535.
Voce puo' valere: 1 per la voce 1
2 per la voce 2
3 per il rumore bianco della voce 2
Nell'esecuzione di due comandi SOUND successivi con voce uguale, il secondo inizia quando il primo suono e' terminato; ponendo la durata 0 si ottiene di far cessare immediatamente il suono selezionato.

==========

SPC — Funzione di stampa
Immediato/Programma

SPC(esp)
dove esp deve dare un risultato numerico compreso tra 0 e 255, di cui viene considerata solo la parte intera.
La funzione provoca il salto di n posizioni, se n e' il valore di esp, anche con passaggio alle linee successive, cioe' corrisponde all'invio di n caratteri "cursore a destra". L'uso di SPC consente di ottenere incolonnamenti, tenendo conto anche della lunghezza dei dati.

==========

SQR — Funzione numerica
Immediato/Programma

SQR(esp)
dove esp deve essere un'espressione numerica maggiore o uguale a zero. Fornisce la radice quadrata di esp.

==========

SSHAPE — Istruzione
Immediato/Programma

SSHAPE var,x1,y1 [,x2,y2]
dove:
var e' il nome di una variabile stringa
x1,y1 sono le coordinate di un angolo della zona video da memorizzare
x2,y2 sono le coordinate dell'angolo opposto
L'istruzione, che ha un effetto immediatamente visibile solo se usata in uno dei modi grafici, consente di memorizzare zone rettangolari del video in una stringa. E' necessario calcolare le dimensioni che deve assumere la stringa, che non puo' superare i 255 caratteri, e, in caso, delimitare la zona o frazionarla per memorizzarla in piu' strighe. Per calcolare le

dimensioni si possono usare le formule che seguono:

per l'alta risoluzione

```
INT((ABS(x1-x2)+1)/4+.99)*(ABS(y1-y2)+1)+4
```

per il multicolore

```
INT((ABS(x1-x2)+1)/8+.99)*(ABS(y1-y2)+1)+4
```

La memorizzazione avviene riga per riga; negli ultimi 4 byte della stringa si trovano le lunghezze delle colonne e delle righe del rettangolo, nel formato byte basso-byte alto. Tali lunghezze, se e' attivo SCALE, devono essere divise rispettivamente per 5.12 (colonne) e 3.2 (righe).

==========

STEP — Istruzione Immediato/Programma

STEP esp

dove esp deve essere un'espressione numerica e rappresenta l'incremento da usare nel controllo di un ciclo con l'istruzione FOR. Si veda FOR.

==========

STOP — Istruzione Immediato/Programma

STOP

ferma il programma e compare il messaggio indicante il numero di linea della STOP. Ha lo stesso effetto della pressione del tasto RUN/STOP. Per continuare premere CONT, se e' possibile continuare.

==========

STR$ — Funzione stringa Immediato/Programma

STR$(esp)

dove esp deve essere un'espressione numerica. Fornisce la stringa corrispondente al numero

risultato di esp. La stringa generata contiene all'inizio o uno spazio o il segno meno.

==========

SYS — Istruzione Immediato/Programma

SYS ind
dove ind e' un indirizzo di memoria (compreso tra 0 e 65535). All'indirizzo indicato deve esserci un sottoprogramma in linguaggio macchina che va in esecuzione; esso deve terminare con il comando RTS che fa proseguire il programma BASIC dall'istruzione successiva alla SYS. A differenza di USR, con SYS non si passano parametri; per questa ragione i risultati del programma in linguaggio macchina devono essere memorizzati in posizioni di memoria definite in modo opportuno, e poi da li' prelevati con la funzione PEEK.

==========

TAB — Funzione di stampa Immediato/Programma

TAB(esp)
dove esp deve fornire un valore numerico compreso tra 0 e 255. Per il video esp rappresenta la posizione di stampa dove portare il cursore contando dall'inizio della linea; se la posizione e' gia' stata superata dal cursore, essa non agisce; se la posizione e' fuori dalla linea, prosegue sulle linee seguenti.
Quando la stampa e' diretta alla stampante la funzione TAB agisce come la funzione SPC, cioe' il conto della posizione e' relativo alla posizione attuale.

==========

TAN — Funzione numerica
Immediato/Programma

TAN(x)
fornisce la tangente dell'angolo x, che deve essere espresso in radianti.

==========

TO — Istruzione
Immediato/Programma

TO
fa parte delle istruzioni: BACKUP, COPY, DRAW,FOR, GO TO, e a queste si rimanda.

==========

TRAP — Istruzione
Immediato/Programma

TRAP [n]
consente di rendere attiva una routine di errore, scritta dall'utente a partire dalla linea n. In tale routine possono essere analizzate le variabili EL, ER, ERR$ per conoscere quale errore si e' verificato ed agire in conseguenza. Per uscire dalla routine di errore si deve usare l'istruzione RESUME. Quando e' stata eseguita l'istruzione TRAP n, va in esecuzione la routine di errore per tutti gli errori salvo quello di codice 17 "UNDEF'D STATEMENT ERROR". Va in esecuzione la routine anche se si preme il tasto RUN/STOP.
L'uso di TRAP senza parametro disabilita la funzione.

==========

TROFF — Istruzione
Immediato/Programma

TROFF
disabilita la funzione TRON e non vengono piu'

listati i numeri delle linee di programma eseguite.

==========

TRON Istruzione
Immediato/Programma

TRON
abilita la stampa dei numeri di linea eseguiti dal programma. E' molto utile per trovare errori nei programmi. I numeri di linea compaiono tra parentesi quadre.

==========

UNTIL Istruzione
Immediato/Programma

UNTIL
fa parte dell'istruzione DO...LOOP, a cui si rimanda

==========

USR Funzione numerica
Immediato/Programma

USR(x)
consente di andare ad eseguire un programma in linguaggio macchina, il cui indirizzo di inizio deve essere stato precedentemente memorizzato nei byte 1281 e 1282 (byte basso-byte alto). Il parametro x viene passato al programma in linguaggio macchina nell'accumulatore 1; esso e' formato da una serie di 5 byte da 97 a 102, cosi' utilizzati: 97 per l'esponente, 98-101 per la mantissa e 102 per il segno. Se il parametro e', invece, una stringa: 97 per il numero dei caratteri, 98-99 per il puntatore al primo carattere del corpo della stringa. In tale accumulatore si trova un eventuale risultato, che pero' viene anche reso disponibile al programma BASIC come valore della funzione. Il

programma in linguaggio macchina deve terminare con l'istruzione RTS.

==========

VAL — Funzione numerica
Immediato/Programma

VAL(a$)
dove a$ puo' essere una costante o una variabile stringa e deve essere di contenuto numerico. La funzione trasforma la stringa in un numero. La conversione si ferma al primo carattere non numerico incontrato. Se il primo carattere della stringa non e' numerico (+, - o cifra) il risultato e' 0.

==========

VERIFY — Comando
Immediato/Programma

VERIFY "nomef"[,unita'][,flag]
confronta il programma presente in memoria con quello selezionato o su disco o su nastro. E' buona norma eseguire la verifica dei programmi dopo averli memorizzati. Nel caso della cassetta questa istruzione puo' essere usata per far avanzare il nastro fino a dopo l'ultimo programma memorizzato; basta usare VERIFY con quel nome e naturalmente il risultato non sara' O.K., ma il nastro sara' arrivato al punto giusto. Il flag deve essere presente, quando necessario, se il programma da verificare e' stato memorizzato con il flag; a questo proposito vale quanto detto per il flag nella descrizione del comando SAVE.

==========

VOL — Istruzione
Immediato/Programma

VOL n
predispone il volume per il comando SOUND. Il

numero n puo' variare tra 0 e 8. Il valore 0 annulla il volume. Il volume e' il medesimo per le due voci.

==========

WAIT Istruzione
Immediato/Programma

WAIT n,x1 [,x2]
dove:
n e' un indirizzo di memoria, da 0 a 65535
x1 e' un numero tra 0 e 255
x2 e' un numero tra 0 e 255
L'operazione agisce cosi':
. viene analizzato il contenuto del byte n,
. viene eseguita una operazione di OR-esclusivo (XOR) tra il contenuto del byte di indirizzo n e il numero x2; se x2 manca viene assunto uguale a 0,
. viene eseguita una operazione AND tra il risultato precedente e x1,
. se il risultato finale e' 0, cioe' FALSO, viene eseguita nuovamente la WAIT,
. se il risultato e' diverso da zero, cioe' VERO, il programma prosegue.
Con l'uso non appropriato di questa istruzione si possono produrre dei cicli infiniti.
Le regole delle operazioni logiche AND e XOR sono le seguenti:

```
1 AND 1 = 1     1 AND 0 = 0
0 AND 1 = 0     0 AND 0 = 0

1 XOR 1 = 0     1 XOR 0 = 1
0 XOR 1 = 1     0 XOR 0 = 0
```

==========

WHILE Istruzione
Immediato/Programma

WHILE
fa parte dell'istruzione DO...LOOP, a cui si rimanda.

APPENDICE B

CODICI E NUMERI DEL CALCOLATORE

La memoria del calcolatore e' formata da celle che vengono chiamate BYTE. Ogni byte e' contraddistinto da un INDIRIZZO numerico che puo' variare da 0 a 65535, come indicato in Figura B.1.

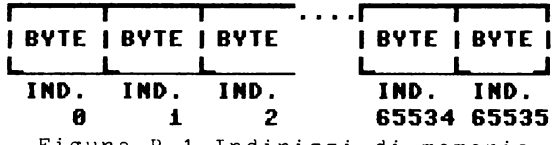
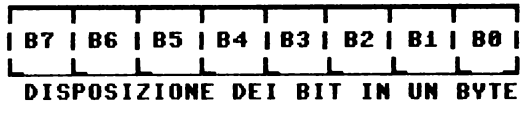


Figura B.1 Indirizzi di memoria

Ogni BYTE e' formato da 8 BIT, cifre binarie che possono essere 0 o 1, come schematizzato in Figura B.2.



DISPOSIZIONE DEI BIT IN UN BYTE

NOTA: BI SIGNIFICA BIT DI POSIZIONE I
B0 E' IL BIT MENO SIGNIFICATIVO
B7 E' IL BIT PIU' SIGNIFICATIVO

Figura B.2 Disposizione dei BIT in un BYTE

In ogni BYTE si puo' rappresentare un numero binario formato da 8 BIT (0 o 1); i pesi di ogni cifra sono quelli indicati in Figura B.3.

```
POS.  7    6    5    4    3    2    1    0
     ┌────┬────┬────┬────┬────┬────┬────┬────┐
     |0/1|0/1|0/1|0/1|0/1 0/1 0/1 0/1
     └────┴────┴────┴────┴────┴────┴────┴────┘
       7    6    5    4    3    2    1    0
PESI  2    2    2    2    2    2    2    2

NOTA: B0 HA PESO   1   B1 HA PESO   2
      B2 HA PESO   4   B3 HA PESO   8
      B4 HA PESO  16   B5 HA PESO  32
      B6 HA PESO  64   B7 HA PESO 128
```

Figura B.3 Pesi dei BIT in un BYTE

Per calcolare il valore decimale N del numero contenuto in un byte si deve usare la formula che segue, dove "bi" significa "bit di posizione i" e "2^i" signica "2 elevato a i":

N=b7*2^7+b6*2^6+b5*2^5+b4*2^4+b3*2^3+b2*2^2+b1*2^1+b0*2^0
cioe':
N=b7*128+b6*64+b5*32+b4*16+b3*8+b2*4+b1*2+b0*1.

Si ottiene per N, il valore 0 quando tutti i bit sono 0, e il valore 255 quando tutti i bit sono 1, cioe' 256 numeri diversi.

L'interpretazione che viene data al numero contenuto in un byte dipende dal contesto nel quale esso viene preso in esame. Durante la lista (LIST) di un programma il numero contenuto in un byte puo' dar luogo a una parola chiave del linguaggio BASIC. Uno o piu' byte contigui possono rappresentare un numero reale espresso in forma esponenziale (floating-point) o un numero intero. Una serie contigua di byte

puo' rappresentare una parola codificata carattere per carattere.

I CODICI

Per rappresentare i caratteri nella memoria del calcolatore o nei supporti magnetici di memorizzazione, come dischi o nastri, o per inviare dati alla stampante il sistema usa il codice CBM-ASCII, occupando un byte per ogni carattere. In questo codice sono disponibili 256 caratteri diversi, codificati da 0 a 255. Non tutti questi caratteri sono stampabili; alcuni sono usati come codici di controllo per produrre effetti particolari.
Il COMMODORE 16 dispone di due gruppi diversi di codici ASCII, si dice di due SET di caratteri; il primo si chiama SET MAIUSCOLO/GRAFICO e il secondo SET MINUSCOLO/MAIUSCOLO. E' possibile usare i due SET di caratteri in alternativa, mai contemporaneamente.
Per evidenziare i caratteri sul video (cioe' per scrivere o disegnare) il sistema usa un codice leggermente diverso dal codice ASCII, ma con esso relazionato, che si chiama D-CODE.
Analizziamo i modi che l'utente ha per inviare dati da tastiera al calcolatore:
.1) Il calcolatore e' in stato comandi, si possono scrivere comandi in immediato o istruzioni di programma, le parole chiave del linguaggio e i simboli sono trasformati in TOKENS, come indicato in Tabella B.2; le costanti numeriche o stringa, i nomi delle variabili e delle funzioni utente sono memorizzate in codice ASCII, come indicato in Tabella B.1.
.2) Il calcolatore e' in stato programma e esegue una istruzione di richiesta dati da tastiera; esso accetta caratteri codificati ASCII, che corrispondono ai tasti premuti, e li trasforma nel formato richiesto dal tipo di variabile alla quale sono destinati, con eventuale segna-

lazione di errore. Gli errori possibili sono:
..variabile numerica contro dato alfabetico,
..variabile stringa contro dato piu' lungo di 255 caratteri.
In ambedue gli stati indicati il sistema fa comparire quello che si scrive, carattere per carattere, sul video nella posizione attuale del cursore.
Riepiloghiamo i modi che ha l'utente per scrivere da programma sul video:
.1) Usare l'istruzione PRINT seguita dal nome di una o piu' variabili e/o da una o piu' costanti separate dai separatori consentiti. Il sistema fa comparire i dati carattere per carattere a partire dalla posizione attuale del cursore; eventuali caratteri di controllo presenti possono spostare la posizione del cursore. I caratteri appaiono del colore attivo per il testo. Usando i codici di controllo del colore si puo' modificare il colore dei caratteri.
La lista dei dati da stampare, che segue la parola chiave PRINT, puo' comprendere anche funzioni, come, per esempio, la CHR$. Se l'argomento di CHR$ e' il codice di un carattere stampabile, esso compare, se, invece, esso e' un codice di controllo, produce il suo effetto.
.2) Usare l'istruzione CHAR che serve per scrivere una stringa di caratteri in una definita posizione del video.
.3) Usare due istruzioni POKE per memorizzare in una posizione della MAPPA VIDEO il D-CODE di un carattere e nella corrispondente posizione della MAPPA COLORE il codice del colore desiderato, che deve essere diverso dal colore dello sfondo. Questi argomenti saranno trattati nel secondo volume; ci limitiamo a dire che al video corrisponde una zona di memoria (MAPPA VIDEO) che puo' contenere i codici (D-CODE) dei 1000 caratteri visualizzabili, e una zona di memoria (MAPPA COLORE) per i codici del colore che deve avere ogni carattere.

Nei casi 1 e 2 e' il sistema che esegue le due POKE necessarie per evidenziare un carattere sul video.

I caratteri stampabili di ogni SET sono 128; essi pero' possono essere evidenziati in due modi, positivo e negativo (si dice in campo diretto e in campo inverso, cioe' scambiando tra loro il colore dello sfondo e il colore del carattere). I D-CODE dei caratteri stampabili diretti vanno da 0 a 127, quelli dei caratteri inversi, da 128 a 255.

Nella Tabella B.1 sono riportati tutti i caratteri stampabili, sia diretti che inversi, per i due SET di caratteri disponibili, i codici ASCII, espressi in decimale, e i D-CODE dei caratteri diretti e dei caratteri inversi, espressi in decimale. Usando la funzione CHR$ con argomento uguale al codice decimale ASCII si ottiene la stampa del carattere diretto. Per ottenere il carattere inverso si deve usare il codice di controllo necessario per attivare RVS-ON (CHR$(18)). Quando invece si scrive con l'istruzione POKE si puo' usare il D-CODE del carattere inverso.

CORRISP. CARATTERI, CODICI ASCII, D/CODE PER I DUE SET DISPONIBILI

MA/GRAF		MIN/MA		ASCII	D/CODE	
DIR.	INV.	DIR.	INV.	DEC.	DIR.	INV.
	■		■	32	32	160
!	▙	!	▙	33	33	161
"	▜	"	▜	34	34	162
#	⠿	#	⠿	35	35	163
$	⚅	$	⚅	36	36	164
%	▨	%	▨	37	37	165
&	⚁	&	⚁	38	38	166

TABELLA B.1 Caratteri, codice ASCII e D-CODE

CORRISP. CARATTERI, CODICI ASCII, D/CODE
PER I DUE SET DISPONIBILI

MA/GRAF		MIN/MA		ASCII	D/CODE	
DIR.	INV.	DIR.	INV.	DEC.	DIR.	INV.
'	▚	'	▚	39	39	167
(	▌	(	▌	40	40	168
)	▐	)	▐	41	41	169
*	▞	*	▞	42	42	170
+	▛	+	▛	43	43	171
,	▜	,	▜	44	44	172
-	▬	-	▬	45	45	173
.	▖	.	▖	46	46	174
/	▗	/	▗	47	47	175
0	0	0	0	48	48	176
1	1	1	1	49	49	177
2	2	2	2	50	50	178
3	3	3	3	51	51	179
4	4	4	4	52	52	180
5	5	5	5	53	53	181
6	6	6	6	54	54	182
7	7	7	7	55	55	183
8	8	8	8	56	56	184
9	9	9	9	57	57	185
:	▘	:	▘	58	58	186
;	▝	;	▝	59	59	187
<	▚	<	▚	60	60	188
=	▞	=	▞	61	61	189
>	▟	>	▟	62	62	190
?	▜	?	▜	63	63	191
@	@	@	@	64	0	128
A	A	a	a	65	1	129
B	B	b	b	66	2	130
C	C	c	c	67	3	131
D	D	d	d	68	4	132
E	E	e	e	69	5	133

TABELLA B.1 Caratteri, codice ASCII e D-CODE
(continuazione)

CORRISP. CARATTERI, CODICI ASCII, D/CODE PER I DUE SET DISPONIBILI

MA/GRAF		MIN/MA		ASCII	D/CODE	
DIR.	INV.	DIR.	INV.	DEC.	DIR.	INV.
F	F	f	f	70	6	134
G	G	g	g	71	7	135
H	H	h	h	72	8	136
I	I	i	i	73	9	137
J	J	j	j	74	10	138
K	K	k	k	75	11	139
L	L	l	l	76	12	140
M	M	m	m	77	13	141
N	N	n	n	78	14	142
O	O	o	o	79	15	143
P	P	p	p	80	16	144
Q	Q	q	q	81	17	145
R	R	r	r	82	18	146
S	S	s	s	83	19	147
T	T	t	t	84	20	148
U	U	u	u	85	21	149
V	V	v	v	86	22	150
W	W	w	w	87	23	151
X	X	x	x	88	24	152
Y	Y	y	y	89	25	153
Z	Z	z	z	90	26	154
[	[	[	[	91	27	155
£	£	£	£	92	28	156
]	]	]	]	93	29	157
↑	↑	↑	↑	94	30	158
←	←	←	←	95	31	159
—	—	—	—	96	64	192
♠	♠	A	A	97	65	193
|	|	B	B	98	66	194
—	—	C	C	99	67	195
—	—	D	D	100	68	196
—	—	E	E	101	69	197
—	—	F	F	102	70	198
|	|	G	G	103	71	199

TABELLA B.1 Caratteri, codice ASCII e D-CODE (continuazione)

CORRISP. CARATTERI, CODICI ASCII, D/CODE
PER I DUE SET DISPONIBILI

MA/GRAF		MIN/MA		ASCII	D/CODE	
DIR.	INV.	DIR.	INV.	DEC.	DIR.	INV.
		H		104	72	200
		I		105	73	201
		J		106	74	202
		K		107	75	203
		L		108	76	204
		M		109	77	205
		N		110	78	206
		O		111	79	207
		P		112	80	208
		Q		113	81	209
		R		114	82	210
		S		115	83	211
		T		116	84	212
		U		117	85	213
		V		118	86	214
		W		119	87	215
		X		120	88	216
		Y		121	89	217
		Z		122	90	218
		+		123	91	219
				124	92	220
				125	93	221
				126	94	222
				127	95	223
				160	96	224
				161	97	225
				162	98	226
				163	99	227
				164	100	228
				165	101	229
				166	102	230
				167	103	231

TABELLA B.1 Caratteri, codice ASCII e D-CODE
(continuazione)

CORRISP. CARATTERI, CODICI ASCII, D/CODE PER I DUE SET DISPONIBILI

MA/GRAF DIR.	MA/GRAF INV.	MIN/MA DIR.	MIN/MA INV.	ASCII DEC.	D/CODE DIR.	D/CODE INV.
				168	104	232
				169	105	233
				170	106	234
				171	107	235
				172	108	236
				173	109	237
				174	110	238
				175	111	239
				176	112	240
				177	113	241
				178	114	242
				179	115	243
				180	116	244
				181	117	245
				182	118	246
				183	119	247
				184	120	248
				185	121	249
				186	122	250
				187	123	251
				188	124	252
				189	125	253
				190	126	254
				191	127	255
				192	64	192
		A		193	65	193
		B		194	66	194
		C		195	67	195
		D		196	68	196
		E		197	69	197
		F		198	70	198
		G		199	71	199
		H		200	72	200

TABELLA B.1 Caratteri, codice ASCII e D-CODE (continuazione)

CORRISP. CARATTERI, CODICI ASCII, D/CODE
PER I DUE SET DISPONIBILI

MA/GRAF		MIN/MA		ASCII	D/CODE	
DIR.	INV.	DIR.	INV.	DEC.	DIR.	INV.
		I		201	73	201
		J		202	74	202
		K		203	75	203
		L		204	76	204
		M		205	77	205
		N		206	78	206
		O		207	79	207
		P		208	80	208
		Q		209	81	209
		R		210	82	210
		S		211	83	211
		T		212	84	212
		U		213	85	213
		V		214	86	214
		W		215	87	215
		X		216	88	216
		Y		217	89	217
		Z		218	90	218
		+		219	91	219
				220	92	220
				221	93	221
				222	94	222
				223	95	223
				224	96	224
				225	97	225
				226	98	226
				227	99	227
				228	100	228
				229	101	229
				230	102	230
				231	103	231
				232	104	232
				233	105	233
				234	106	234

TABELLA B.1 Caratteri, codice ASCII e D-CODE
(continuazione)

CORRISP. CARATTERI, CODICI ASCII, D/CODE PER I DUE SET DISPONIBILI

MA/GRAF DIR.	MA/GRAF INV.	MIN/MA DIR.	MIN/MA INV.	ASCII DEC.	D/CODE DIR.	D/CODE INV.
├	╟	├	╟	235	107	235
▪	▛	▪	▛	236	108	236
└	╙	└	╙	237	109	237
┐	▟	┐	▟	238	110	238
_	▄	_	▄	239	111	239
┌	▞	┌	▞	240	112	240
┴	╩	┴	╩	241	113	241
┬	▀	┬	▀	242	114	242
┤	▌	┤	▌	243	115	243
│	▌	│	▌	244	116	244
▌	▌	▌	▌	245	117	245
▌	▌	▌	▌	246	118	246
─	═	─	═	247	119	247
▬	▬	▬	▬	248	120	248
▬	▬	▬	▬	249	121	249
┘	▪	╱	◪	250	122	250
▪	▜	▪	▜	251	123	251
▪	▙	▪	▙	252	124	252
┘	╝	┘	╝	253	125	253
▪	▟	▪	▟	254	126	254
▫	▣	※	※	255	94	222

TABELLA B.1 Caratteri, codice ASCII e D-CODE (continuazione)

La Tabella B.1 si riferisce ai soli caratteri stampabili, che sono 128 diretti e 128 inversi; le prime due colonne riportano i caratteri del SET MAIUSCOLO/GRAFICO, le due colonne successive quelli del SET MINUSCOLO/MAIUSCOLO.
I codici ASCII vanno da 32 a 127 e da 160 a 255; in realta' i codici da 192 a 223 producono gli stessi caratteri dei codici da 96 a 127, i codici da 224 a 254 producono gli stessi caratteri dei codici da 160 a 190, e il codice 255 produce lo stesso carattere del codice 126.

La Tabella B.2 riguarda i TOKENS, cioe' i codici corrispondenti alle parole chiave e ai simboli del BASIC. In essa la prima colonna riporta il codice, la seconda la parola chiave o il simbolo corrispondente, la terza il modo, se esiste, per abbreviare la scrittura, la quarta l'effetto prodotto sul video dall'abbreviazione. Nella Tabella B.2 e' stato seguito quasi l'ordine alfabetico delle parole chiave, ponendo in fondo i simboli di tipo aritmetico. Le eccezioni all'ordine alfabetico delle parole chiave si hanno per:

..FN, che segue DEF, dato che si usano insieme,

..THEN e ELSE che seguono IF, dato che sono usate insieme a IF,

..USING che segue PRINT e PRINT#, dato che viene usato insieme ad esse.

T O K E N S

CODICE	KEYWORD	ABBREVIAZIONE	VISUALIZZAZIONE
182	ABS	A SHIFT-B	A|
175	AND	A SHIFT-N	A/
198	ASC	A SHIFT-S	A♥
193	ATN	A SHIFT-T	A|
220	AUTO	A SHIFT-U	A,
246	BACKUP	B SHIFT-A	B♠
225	BOX	B SHIFT-O	B┌
224	CHAR	CH SHIFT-A	CH♠
199	CHR$	C SHIFT-H	C|
226	CIRCLE	C SHIFT-I	C\
160	CLOSE	CL SHIFT-O	CL┌
156	CLR	C SHIFT-L	CL
157	CMD	C SHIFT-M	C\
243	COLLECT	COL SHIFT-L	COL

TABELLA B.2 Parole chiave, TOKENS e abbreviazioni

T O K E N S

CODICE	KEYWORD	ABBREVIAZIONE	VISUALIZZAZIONE
231	COLOR	CO SHIFT-L	COL
154	CONT	C SHIFT-O	C┌
244	COPY	CO SHIFT-P	CO┐
190	COS	MANCA	MANCA
131	DATA	D SHIFT-A	D♠
209	DEC	MANCA	MANCA
150	DEF	D SHIFT-E	D─
165	FN	MANCA	MANCA
247	DELETE	DE SHIFT-L	DEL
134	DIM	D SHIFT-I	D↖
238	DIRECTORY	DI SHIFT-R	DI_
240	DLOAD	D SHIFT-L	DL
235	DO	MANCA	MANCA
229	DRAW	D SHIFT-R	D_
239	DSAVE	D SHIFT-S	D♥
128	END	E SHIFT-N	E/
211	ERR$	E SHIFT-R	E_
237	EXIT	EX SHIFT-I	EX↖
189	EXP	E SHIFT-X	E↑
129	FOR	F SHIFT-O	F┌
184	FRE	F SHIFT-R	F_
161	GET	G SHIFT-E	G─
203	GO	MANCA	MANCA
141	GOSUB	GO SHIFT-S	GO♥
137	GOTO	G SHIFT-O	G┌
222	GRAPHIC	G SHIFT-R	G_
227	GSHAPE	G SHIFT-S	G♥
241	HEADER	HE SHIFT-A	HE♠
234	HELP	HE SHIFT-L	HEL
210	HEX$	H SHIFT-E	H─
139	IF	MANCA	MANCA
167	THEN	T SHIFT-H	T|
213	ELSE	E SHIFT-L	EL
133	INPUT	MANCA	MANCA
132	INPUT#	I SHIFT-N	I/
212	INSTR	IN SHIFT-S	IN♥

TABELLA B.2 Parole chiave, TOKENS e abbreviazioni (continuazione)

T O K E N S

CODICE	KEYWORD	ABBREVIAZIONE	VISUALIZZAZIONE
181	INT	MANCA	MANCA
207	JOY	J SHIFT-O	J┌
249	KEY	K SHIFT-E	K─
200	LEFT$	LE SHIFT-F	LE└
195	LEN	MANCA	MANCA
136	LET	L SHIFT-E	L─
155	LIST	L SHIFT-I	L╲
147	LOAD	L SHIFT-O	L┌
230	LOCATE	LO SHIFT-C	LO└
188	LOG	MANCA	MANCA
236	LOOP	LO SHIFT-O	LO┌
202	MID$	M SHIFT-I	M╲
250	MONITOR	M SHIFT-O	M┌
162	NEW	MANCA	MANCA
130	NEXT	N SHIFT-E	N─
168	NOT	N SHIFT-O	N┌
145	ON	MANCA	MANCA
159	OPEN	O SHIFT-P	O┐
176	OR	MANCA	MANCA
223	PAINT	P SHIFT-A	P♠
194	PEEK	P SHIFT-E	P─
151	POKE	P SHIFT-O	P┌
185	POS	MANCA	MANCA
153	PRINT	?	?
152	PRINT#	P SHIFT-R	P└
251	USING	US SHIFT-I	US╲
221	PUDEF	P SHIFT-U	P╱
205	RCLR	R SHIFT-C	R─
208	RDOT	R SHIFT-D	R─
135	READ	R SHIFT-E	R─
143	REM	MANCA	MANCA
245	RENAME	RE SHIFT-N	RE/
248	RENUMBER	REN SHIFT-U	REN╱
140	RESTORE	RE SHIFT-S	RE♥
214	RESUME	RES SHIFT-U	RES╱
142	RETURN	RE SHIFT-T	RE|
204	RGR	R SHIFT-G	R|
201	RIGHT$	R SHIFT-I	R╲

TABELLA B.2 Parole chiave, TOKENS e abbreviazioni (continuazione)

T O K E N S

CODICE	KEYWORD	ABBREVIAZIONE	VISUALIZZAZIONE
206	RLUM	R SHIFT-L	RL
187	RND	R SHIFT-N	R/
138	RUN	R SHIFT-U	R,
148	SAVE	S SHIFT-A	S♠
233	SCALE	SC SHIFT-A	SC♠
232	SCNCLR	S SHIFT-C	S—
242	SCRATCH	SC SHIFT-R	SC_
180	SGN	S SHIFT-G	S|
191	SIN	S SHIFT-I	S↘
218	SOUND	S SHIFT-O	S┌
166	SPC(	S SHIFT-P	S┐
186	SQR	S SHIFT-Q	S●
228	SSHAPE	S SHIFT-S	S♥
169	STEP	ST SHIFT-E	ST—
144	STOP	S SHIFT-T	S|
196	STR$	ST SHIFT-R	ST_
158	SYS	S SHIFT-Y	S |
163	TAB(	T SHIFT-A	T♠
192	TAN	MANCA	MANCA
164	TO	MANCA	MANCA
215	TRAP	T SHIFT-R	T_
217	TROFF	TRO SHIFT-F	TRO—
216	TRON	TR SHIFT-O	TR┌
252	UNTIL	U SHIFT-N	U/
183	USR	U SHIFT-S	U♥
197	VAL	V SHIFT-A	V♠
149	VERIFY	V SHIFT-E	V—
219	VOL	V SHIFT-O	V┌
146	WAIT	W SHIFT-A	W♠
253	WHILE	W SHIFT-H	W|
170	+	+	+
171	-	-	-
172	*	*	*
173	/	/	/
174	↑	↑	↑
177	>	>	>
178	=	=	=
179	<	<	<

TABELLA B.2 Parole chiave, TOKENS e abbreviazioni (continuazione)

Nella Tabella B.3 riportiamo l'elenco dei TOKENS in ordine di codice crescente.

TOKENS ORDINATI

CODICE	KEYWORD
128	END
129	FOR
130	NEXT
131	DATA
132	INPUT#
133	INPUT
134	DIM
135	READ
136	LET
137	GOTO
138	RUN
139	IF
140	RESTORE
141	GOSUB
142	RETURN
143	REM
144	STOP
145	ON
146	WAIT
147	LOAD
148	SAVE
149	VERIFY
150	DEF
151	POKE
152	PRINT#
153	PRINT
154	CONT
155	LIST
156	CLR
157	CMD
158	SYS
159	OPEN

TABELLA B.3 TOKENS in ordine di codice

CODICE	KEYWORD
160	CLOSE
161	GET
162	NEW
163	TAB(
164	TO
165	FN
166	SPC(
167	THEN
168	NOT
169	STEP
170	+
171	-
172	*
173	/
174	↑
175	AND
176	OR
177	>
178	=
179	<
180	SGN
181	INT
182	ABS
183	USR
184	FRE
185	POS
186	SQR
187	RND
188	LOG
189	EXP
190	COS
191	SIN
192	TAN
193	ATN
194	PEEK
195	LEN
196	STR$
197	VAL

TABELLA B.3 TOKENS in ordine di codice
(continuazione)

CODICE	KEYWORD
198	ASC
199	CHR$
200	LEFT$
201	RIGHT$
202	MID$
203	GO
204	RGR
205	RCLR
206	RLUM
207	JOY
208	RDOT
209	DEC
210	HEX$
211	ERR$
212	INSTR
213	ELSE
214	RESUME
215	TRAP
216	TRON
217	TROFF
218	SOUND
219	VOL
220	AUTO
221	PUDEF
222	GRAPHIC
223	PAINT
224	CHAR
225	BOX
226	CIRCLE
227	GSHAPE
228	SSHAPE
229	DRAW
230	LOCATE
231	COLOR
232	SCNCLR
233	SCALE
234	HELP
235	DO

TABELLA B.3 TOKENS in ordine di codice (continuazione)

CODICE	KEYWORD
236	LOOP
237	EXIT
238	DIRECTORY
239	DSAVE
240	DLOAD
241	HEADER
242	SCRATCH
243	COLLECT
244	COPY
245	RENAME
246	BACKUP
247	DELETE
248	RENUMBER
249	KEY
250	MONITOR
251	USING
252	UNTIL
253	WHILE

TABELLA B.3 TOKENS in ordine di codice (continuazione)

Dalla Tabella B.2 sono escluse le parole riservate riportate nella Tabella B.4; esse sono codificate carattere per carattere in codice ASCII. Inoltre, l'istruzione GET# viene memorizzata utilizzanto il codice di GET, seguito dal codice ASCII di #, cioe' come 161 e 35 in due byte.
Per il significato delle variabili della Tabella B.4, rimandiamo all'Appendice A.

PAROLE RISERVATE CODIFICATE IN ASCII	
NOME	**CODIFICA ASCII**
DS	68 83
DS$	68 83 36
EL	69 76
ER	69 82
ERR$	69 82 82 36
ST	83 84
TI	84 73
TI$	84 73 36
π	255

TABELLA B.4 Parole riservate codificate ASCII

Nella Tabella B.5 riportiamo il significato di alcuni dei codici ASCII che vanno da 0 a 31 e da 128 a 160, quando sono usati come argomento della funzione CHR$. Per alcuni codici viene indicato anche l'effetto prodotto relativamente alla stampante MPS801.

C O D I C I D I C O N T R O L L O

CODICI	SIGNIFICATO
5	COLORE BIANCO (CTRL-2)
8	DISABILITA SHIFT-CBM
	MPS801 ATTIVA MODO GRAFICO PER PUNTI
9	ABILITA SHIFT-CBM
10	MPS801 MANDA LINE FEED
13	RETURN
	MPS801 RETURN + LINE FEED
14	ATTIVA SET MAIUSC./MINUSC.
	MPS801 ATTIVA DOPPIA AMPIEZZA
15	MPS801 DISABILITA COD. 14
16	MPS801 SPOSTAMENTO POS.
17	EFFETTO DI FRECCIA GIU'
	MPS801 ATTIVA SET MINUSC./MAIUSC.
18	ATTIVA RVS ON
	MPS801 ATTIVA RVS ON
19	CURSORE IN POSIZIONE HOME
20	CANCELLA (DELETE)
26	MPS801 RIPETE CAR. GRAFICI
27	FUNZIONE DI ESCAPE
	MPS801 SPOSTA A UNA POSIZIONE PUNTO
28	COLORE ROSSO (CTRL-3)
29	EFFETTO FRECCIA A DESTRA
30	COLORE VERDE (CTRL-6)
31	COLORE BLU (CTRL-7)
129	COLORE ARANCIONE (CBM-1)
130	FLASH ON
131	FLASH OFF

TABELLA B.5 Codici di controllo

CODICI	SIGNIFICATO
141	SHIFT-RETURN
142	ATTIVA SET MAIUSC./GRAF.
144	COLORE NERO (CTRL-1)
145	EFFETTO FRECCIA SU MPS801 ATTIVA SET MAIUSC./GRAF.
146	DISATTIVA RVS ON (RVS OFF)
147	PULIZIA VIDEO E CURSORE IN POS. HOME
148	ATTIVA INSERIMENTO (INS)
149	COLORE BRUNO (CBM-2)
150	COL. GIALLO VERDE (CBM-3)
151	COLORE ROSA (CBM-4)
152	COLORE BLU VERDE (CBM-5)
153	COLORE BLU CHIARO (CBM-6)
154	COLORE BLU SCURO (CBM-7)
155	COL. VERDE CHIARO (CBM-8)
156	COLORE PORPORA (CTRL-5)
157	EFFETTO FRECCIA A SINISTRA
158	COLORE GIALLO (CTRL-8)
159	COLORE CIANO (CTRL-4)

TABELLA B.5 Codici di controllo (continuazione)

Nella Tabella B.6 sono indicate le relazioni tra i codici ASCII, indicati con A, e i D-CODE, indicati con D.

RELAZIONE TRA CODICE ASCII E D-CODE	
ASCII	D-CODE
32<=A<= 63	D = A
64<=A<= 95	D = A - 64
96<=A<=127	D = A - 32
160<=A<=191	D = A - 64
192<=A<=254	D = A -128
A =255	D = A -161

TABELLA B.6 Relazione tra codice ASCII e D-CODE

Riepiloghiamo i modi disponibili per cambiare il set di caratteri.

.1) Premere contemporaneamente i tasti SHIFT e CBM fa passare dal set attivo all'altro. La pressione di questi due tasti non ha effetto se e' stato prima eseguito il comando: PRINT CHR$(8), che disabilita i tasti SHIFT-CBM. Nel caso basta eseguire il comando: PRINT CHR$(9) per abilitare nuovamente i tasti SHIFT-CBM.

.2) Eseguire il comando: PRINT CHR$(142) per attivare il set maiuscolo/grafico. Eseguire il comando PRINT CHR$(14) per attivare il set minuscolo/maiuscolo.

I caratteri, che vengono evidenziati sul video, sono descritti per punti in una zona di memoria ROM del calcolatore. Ogni carattere e' rappresentato da 64 punti disposti in una matrice di 8 righe e 8 colonne; in conseguenza per descrivere un carattere vengono usati 8 byte, cioe' 64 bit. Il primo byte rappresenta i punti della

```
***STAMPA IMMAGINE CARATTERI***

D/CODE= 0  SET MAIUSCOLO/GRAFICO

  CARATTERE        BINARIO        DECIMALE

   ****            00111100         60
  **  **           01100110         102
  ** ***           01101110         110
  ** ***           01101110         110
  **               01100000         96
  **   *           01100010         98
   ****            00111100         60
                   00000000         0
***STAMPA IMMAGINE CARATTERI***

D/CODE= 1  SET MAIUSCOLO/GRAFICO

  CARATTERE        BINARIO        DECIMALE

    **             00011000         24
   ****            00111100         60
  **  **           01100110         102
  ******           01111110         126
  **  **           01100110         102
  **  **           01100110         102
  **  **           01100110         102
                   00000000         0
```

Figura B.4 Descrizione dei caratteri

```
***STAMPA IMMAGINE CARATTERI***

D/CODE= 65  SET MAIUSCOLO/GRAFICO

  CARATTERE        BINARIO        DECIMALE

     *             00001000         8
    ***            00011100         28
   *****           00111110         62
  *******          01111111         127
  *******          01111111         127
    ***            00011100         28
   *****           00111110         62
                   00000000         0
***STAMPA IMMAGINE CARATTERI***

D/CODE= 88  SET MAIUSCOLO/GRAFICO

  CARATTERE        BINARIO        DECIMALE

    **             00011000         24
    **             00011000         24
  **  **           01100110         102
  **  **           01100110         102
    **             00011000         24
    **             00011000         24
   ****            00111100         60
                   00000000         0
```

Figura B.4 Descrizione dei caratteri (continuazione)

prima riga, il secondo quelli della seconda riga, e cosi' via. I bit a 1 corrispondono ai punti da evidenziare nel colore del testo (inchiostro), i bit a 0 ai punti dove lasciare invariato il colore dello sfondo. Quando un carattere deve essere visualizzato sul video, il sistema, usando come puntatore il D-CODE del carattere, va a prelevare la sua descrizione

dalla tabella relativa e lo disegna per punti nella posizione attuale del cursore.
Nella Figura B.4 sono riportati i caratteri corrispondenti ai D-CODE 0, 1 65 e 88 nel set maiuscolo/grafico. Essi sono disegnati ingranditi usando un asterisco in corrispondenza di ogni bit a 1. Inoltre, viene riportato il contenuto binario e il valore decimale degli 8 byte che descrivono il carattere.

I NUMERI

Quando si considera il contenuto di due byte consecutivi si possono presentare i seguenti casi:
..1) I due byte rappresentano un indirizzo usato dal sistema per la gestione del calcolatore. In questo caso il byte di indirizzo minore (il primo) e' il byte basso (LO) e quello successivo (il secondo) e' il byte alto (HI). Per calcolare il numero rappresentato, se N e' l'indirizzo del primo byte, si procede cosi':

```
Numero = 256*PEEK(N+1) + PEEK(N)
```

..2) I due byte rappresentano un numero intero con segno. In questo caso il primo byte contiene la parte piu' significativa del numero e il secondo la parte meno significativa. Il bit di sinistra del primo byte e' usato per il segno; 0 per numeri positivi e 1 per numeri negativi.
Il valore del numero intero positivo puo' variare da 0 a 32767=127*256+255.
Esempi:
numero decimale: 0
contenuto dei due byte binari:
00000000 00000000

numero decimale: 32767
contenuto dei due byte binari:
01111111 11111111

numero decimale: 837
contenuto dei due byte binari:
00000011 01000101

Il valore del numero intero negativo puo' variare da -32768 a -1.

Per arrivare alla rappresentazione del numero negativo (detta in complemento a 2) si puo' procedere cosi':
..consideriamo la potenza del 2 immediatamente superiore al massimo numero rappresentabile (con 15 bit a disposizione, 8 nel byte basso e 7 nel byte alto, si arriva al massimo al valore posizionale di 2^14), che risulta 2^15, cioe' il numero 32768;
..consideriamo il valore assoluto del numero da rappresentare;
..calcoliamo la differenza tra 32768 (2^15) e il valore assoluto del numero;
..scriviamo in binario il risultato del calcolo precedente;
..alla fine aggiungiamo un bit 1 a sinistra per il segno meno.
Esempio: rappresentazione di -837.
Calcoliamo 32768-837=31931 e troviamo la rappresentazione binaria di 31931 usando il metodo delle sottrazioni successive delle potenze di 2:

31931	-	16384	=	15547	bit	1	in	posizione	14
15547	-	8192	=	7355	bit	1	in	posizione	13
7355	-	4096	=	3259	bit	1	in	posizione	12
3259	-	2048	=	1211	bit	1	in	posizione	11
1211	-	1024	=	187	bit	1	in	posizione	10
187	-	128	=	59	bit	1	in	posizione	7
59	-	32	=	27	bit	1	in	posizione	5
27	-	16	=	11	bit	1	in	posizione	4
11	-	8	=	3	bit	1	in	posizione	3
3	-	2	=	1	bit	1	in	posizione	1
1	-	1	=	0	bit	1	in	posizione	0

il contenuto dei due byte risulta, dopo l'aggiunta del bit 1 di segno in posizione 15:

11111100 10111011

e rappresenta -837.

Si puo' arrivare allo stesso risultato applicando la regola che segue:

..scrivere la rappresentazione binaria del valore assoluto del numero:

00000011 01000101

..sostituire ai bit 1 dei bit 0 e ai bit 0 dei bit 1, e, alla fine sommare un bit 1 in fondo a destra (1+1=10 e 1+0=1 nel calcolo binario):

con la sostituzione si ha 11111100 10111010

con l'aggiunta si ha 11111100 10111011.

Il numero ottenuto e' la rappresentazione in complemento a 2 del numero negativo.

Il numero -1 ha la rappresentazione in complemento a 2: 11111111 11111111, mentre il numero -32768 ha la rappresentazione in complemento a 2: 10000000 00000000.

Con il programma INTERI, che segue, puoi provare a introdurre numeri interi; vedrai alcune cose curiose, come risulta dai risultati che seguono.

```
1 REM INTERI
10 INPUT"NUMERO INTERO: ";N%
20 IFN%=0THENSTOP
30 PRINT"STAMPA NUMERO N%: ";N%
40 GOTO10
```

RISULTATI PROGRAMMA INTERI

```
NUMERO INTERO: ? 32767
STAMPA NUMERO N%:  32767
NUMERO INTERO: ?-32767
STAMPA NUMERO N%: -32767
NUMERO INTERO: ? 45.89
STAMPA NUMERO N%:  45
NUMERO INTERO: ? 32768
STAMPA NUMERO N%: -32768
NUMERO INTERO: ? 50000
STAMPA NUMERO N%: -15536
NUMERO INTERO: ?-88567
STAMPA NUMERO N%: -23031
NUMERO INTERO: ?-32768

?ILLEGAL QUANTITY ERROR IN 10
READY.
```

Come vedi se rispondi con 32768 viene ricevuto -32768; se rispondi con 50000 viene ricevuto -15536, se rispondi con -88567 viene ricevuto -23031. In conseguenza se un programma chiede un numero intero si deve RIGOROSAMENTE rispondere con un numero compreso tra -32767 e +32767. Come hai visto dai risultati del programma INTERI -32768 provoca un messaggio di errore.

Quando un singolo byte rappresenta un numero senza segno, esso varia da 0 a 255. Se si considera il primo bit a sinistra per il segno, allora il numero positivo varia da 0 a 127 e il numero negativo (applicando le regole precedenti al singolo byte) varia da -128 a -1.

I numeri non interi, cioe' i numeri reali, sono rappresentati nella memoria del calcolatore in forma esponenziale (floating-point) occupando 5 byte consecutivi. Il primo dei 5 byte contiene

l'esponente (caratteristica del numero) e gli altri 4 contengono la mantissa, cioe' le cifre significative del numero. Sia la caratteristica che la mantissa sono numeri con segno.
Esempi di numeri decimali espressi in forma esponenziale sono i seguenti:

123456=0.123456*10^6
123466=1.23456*10^5
123456=12.3456*10^4
123456=0.00123456*10^8
123.895=0.123895*10^3
123.895=1.23895*10^2
0.789=789*10^(-3)

Di norma viene detta forma normalizzata quella nella quale la mantissa comincia con la prima cifra significativa preceduta dal punto decimale.
Nella memoria del calcolatore sia la caratteristica che la mantissa sono espressi in binario con segno. Per la caratteristica il campo di variabilita' risulta tra -128 e +127, il che corrisponde a una variabilita' in decimale da 10^(-39) a 10^(+38).
Per il campo di variabilita' della mantissa dobbiamo considerare il valore di 4 byte associati, con il piu' significativo a sinistra che inizia con il bit di segno. Inoltre, dato che la mantissa viene calcolata in modo che il primo bit di sinistra sia sempre 1 (come se il numero rappresentato fosse del tipo 0.1... e si siano trascurati lo 0 e il punto decimale), il sistema trascura il primo bit, che e' sempre 1, e lo aggiunge solo quando il numero viene utilizzato per calcoli o visualizzazione. Questo modo di procedere consente di ampliare l'intervallo di variabilita' del numero. In conseguenza i limiti di variabilita' per la mantissa risultano da -4294967304 a +4294967304. Contrariamente a quanto avviene per gli interi, la mantissa dei numeri reali viene sempre conservata in valore

assoluto e viene aggiunto il primo bit di sinistra a 0 per i positivi e a 1 per i negativi (rappresentazione in grandezza e segno).
I numeri reali vengono stampati con 9 cifre decimali, ma in memoria ne sono conservate 10.

Con il programma REALI, che segue, puoi provare a introdurre numeri qualunque e controllare come vengono stampati.

```
1 REM REALI
10 INPUT"NUMERO REALE: ";N
20 IFN=0THENSTOP
30 PRINT"STAMPA NUMERO N: ";N
40 GOTO10
```

RISULTATI PROGRAMMA REALI

```
NUMERO REALE: ? 9.99999999
STAMPA NUMERO N:  10
NUMERO REALE: ? 999999999
STAMPA NUMERO N:  999999999
NUMERO REALE: ? 1234.56789123
STAMPA NUMERO N:  1234.56789
NUMERO REALE: ? -9898989898
STAMPA NUMERO N: -9.8989899E+09
NUMERO REALE: ? .00000123456789123
STAMPA NUMERO N:  1.23456789E-06
NUMERO REALE: ? 4294967300
STAMPA NUMERO N:  4.2949673E+09
NUMERO REALE: ? 42949673999
STAMPA NUMERO N:  4.2949674E+10
NUMERO REALE: ? -429496736666
STAMPA NUMERO N: -4.29496737E+11
NUMERO REALE: ? 0
STAMPA NUMERO N:  0
```

APPENDICE C

UTILIZZO DELLA MEMORIA

Il COMMODORE 16 utilizza la memoria disponibile con la tecnica dei PUNTATORI. Questo significa che in una zona di memoria RAM, chiamata PAGINA ZERO (corrispondente agli indirizzi piu' bassi), sono memorizzati, in byte di indirizzo prefissato, gli indirizzi di inizio o fine di particolari zone di memoria.

Il programma BASIC dell'utente comincia, di norma, all'indirizzo 4097; l'indirizzo di inizio in memoria del programma BASIC si trova memorizzato nei due byte di indirizzo 43 e 44. Gli indirizzi memorizzati in due byte hanno la parte meno significativa nel primo byte e la piu' significativa nel secondo. Per calcolare l'indirizzo di inizio del programma BASIC si deve eseguire l'istruzione:
I=256*PEEK(44)+PEEK(43)
ottenendo all'accensione I=4097.

Altri indirizzi utili di puntatori sono:

INIZIO VARIABILI E FINE PROGRAMMA BASIC +1: 45 e 46
INIZIO VARIABILI CON INDICE E FINE VARIABILI SINGOLE: 47 e 48
FINE VARIABILI CON INDICE +1: 49 e 50
INDIRIZZO CIMA CORPI STRINGHE: 51 e 52
PUNTATORE CORPO STRINGHE: 53 e 54
INDIRIZZO PIU' ALTO ZONA DEDICATA AL PROGRAMMA UTENTE: 55 e 56.

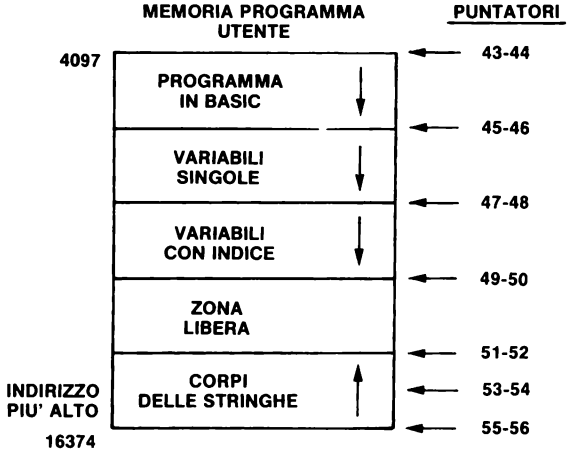


Figura C.1 Utilizzo della memoria e puntatori

Durante la stesura e l'esecuzione di un programma i contenuti di alcuni di questi puntatori cambiano. Se si allunga il programma, le variabili iniziano dopo, se si aggiungono variabili singole si sposta l'inizio delle variabili con indice.

La memoria dedicata al video va da 3072 a 4071. La memoria dedicata ai colori del video va da 2048 a 3047.

Fissiamo la nostra attenzione sulla rappresentazione delle variabili in memoria e utilizziamo alcuni programmi per vedere come esse sono memorizzate.

Iniziamo con il programma INTINMEM. In esso viene chiesto un numero intero e ne viene mostrata la rappresentazione in memoria.

```
1 REM INTINMEM
10 INPUT"▯▯▯NUMERO INTERO = ";N%
15 M$="NUMERI INTERI IN MEMORIA"
20 PRINT"▯▯"M$:PRINT"▯N% = ";N%
25 A=256*PEEK(46)+PEEK(45)
30 FORK=0TO6:PRINTPEEK(A+K);" ";:NEXTK:PRINT
35 STOP
```

Per vedere cosa c'e' in memoria, viene calcolato l'indirizzo di inizio delle variabili; la variabile N% e' la prima definita nel programma e quindi si trova nei primi byte della zona.
Dai risultati vediamo che per una VARIABILE SINGOLA INTERA sono occupati 7 byte consecutivi; i primi due contengono il nome, e precisamente:
.primo byte: codice ASCII prima lettera del nome + 128, per noi 78, codice di N, +128 da' 206,
.secondo byte: 128 solo, dato che la variabile ha un solo carattere.
La costante aggiuntiva 128 consente di distinguere che si tratta di una variabile con suffisso %.
Ai primi due byte ne seguono 5, dei quali pero' sono utilizzati solo i primi 2, nel modo byte alto che precede il byte basso, come puoi vedere dai risultati che seguono.

RISULTATI INTINMEM

```
NUMERI INTERI IN MEMORIA

N% =  0
 206   128   0   0   0   0   0

N% =  255
 206   128   0   255   0   0   0

N% =  256
 206   128   1   0   0   0   0

N% =  513
 206   128   2   1   0   0   0

N% = -1
 206   128   255   255   0   0   0

N% = -256
 206   128   255   0   0   0   0
```

Il programma stampa il contenuto dei 7 byte in decimale (non in binario).

Il programma FLOATINMEM ci permette di vedere come sono memorizzati i numeri reali, cioe' quelli memorizzati in variabili senza suffisso.

```
1 REM FLOATINMEM
10 PRINT"▯▯▯SCRIVI UN NUMERO DECIMALE":INPUTN
15 IFSW=1THEN25
20 P=PEEK(45)+256*PEEK(46):GOSUB235
25 PRINT"▯▯▯▯"C1$;N:L$=""
30 IFPEEK(P)<>78THENSTOP
35 FORI=1TO5:X(I)=PEEK(P+I+1):NEXTI
40 PRINTC2$
45 FORI=2TO5:PRINTX(I);"      ";:NEXTI
50 PRINT:PRINTO$
55 FORI=2TO5:GOSUB165:PRINTF$(I);" ";:NEXTI
60 PRINT
65 PRINTC3$:PRINTL$
70 S=1+2*(LEFT$(F$(2),1)="1")
75 F$(2)="1"+RIGHT$(F$(2),7)
80 PRINTN1$:PRINTN2$
85 LL$=F$(2)+F$(3)+F$(4)+F$(5)
90 PRINTLL$
95 Y=X(1)-128
100 PRINTC8$;X(1)
105 PRINTC5$;Y
110 IFY<0THENM$="0":D$=LEFT$(Z$,-Y)+LL$:GOTO140
115 IFX(1)=0THENM$="0":D$="0":GOTO140
120 IFY>LEN(LL$)THENLL$=LEFT$(LL$+Z$,Y)
125 M$=LEFT$(LL$,Y):Z=LEN(LL$)-Y
130 IFZ<=0THEND$="0":GOTO140
135 D$=RIGHT$(LL$,(LEN(LL$)-Y))
140 PRINTC6$,M$:PRINTC7$,D$
145 GOSUB195:GOSUB215:PRINTC4$;(M+D)*S
150 INPUT"▯▯ANCORA (S/N) ";B$
155 IFB$="S"THENSW=1:GOTO10
160 STOP
165 F$(I)="":FORJ=7TO0STEP-1
170 IFINT(X(I)/2↑J)=0THEN180
175 F$(I)=F$(I)+"1":X(I)=X(I)-2↑J:GOTO185
180 F$(I)=F$(I)+"0"
185 NEXT:L$=L$+F$(I):RETURN
190 PRINTD;" ";
195 M=0:IFM$="0"THENRETURN
200 FORJ=LEN(M$)TO1STEP-1
205 IFMID$(M$,J,1)="1"THENM=M+2↑(LEN(M$)-J)
210 NEXTJ:RETURN
```

```
215 D=0:IFD$="0"THENRETURN
220 FORJ=1TOLEN(D$)
225 IFMID$(D$,J,1)="1"THEND=D+(1/2)↑J
230 NEXTJ:RETURN
235 N1$="STRINGA BIT CON AGGIUNTO BIT INIZIALE"
240 N2$="AL POSTO DEL BIT DI SEGNO"
245 O$="QUATTRO BYTE MANTISSA IN BINARIO"
250 P$="NUMERI FLOATING POINT"
255 C1$="NUMERO INTRODOTTO = "
260 C2$="QUATTRO BYTE MANTISSA IN MEMORIA"
265 C3$="STRINGA DI BIT INIZIALE"
270 C4$="NUMERO CALCOLATO = "
275 C5$="ESP. PER BASE 2 = "
280 C6$="VAL.BIN.PART.INT. SENZA SEGNO"
285 C7$="VAL.BIN.PART.DEC. SENZA SEGNO"
290 C8$="BYTE ESP. = "
295 Z$="0":FORK=1TO127:Z$=Z$+"0":NEXTK
300 RETURN
```

Il programma chiede un numero reale e stampa:
.il numero introdotto,
.i 4 byte della mantissa in decimale,
.i 4 byte della mantissa in binario,
.la stringa di bit dei 4 byte,
.la stringa di bit con aggiunto il bit 1 iniziale al posto del bit di segno,
.il valore decimale dell'esponente,
.l'esponente calcolato per la base 2 (ottenuto sottraendo 128 al valore precedente),
.il valore binario della parte intera senza segno,
.il valore binario della parte non intera,
.il numero ricalcolato.
In certi casi puoi trovare una differenza tra il numero introdotto e quello ricalcolato. La routine che ricalcola il numero, dopo aver aggiunto il bit 1 nella prima posizione a sinistra, si trova da 215 a 230.

RISULTATI FLOATINMEM

```
NUMERO INTRODOTTO =  234.768
QUATTRO BYTE MANTISSA IN MEMORIA
 106        196        155        166
QUATTRO BYTE MANTISSA IN BINARIO
01101010 11000100 10011011 10100110
STRINGA DI BIT INIZIALE
0110101011000100100110111010011 0
STRINGA BIT CON AGGIUNTO BIT INIZIALE
AL POSTO DEL BIT DI SEGNO
1110101011000100100110111010011 0
BYTE ESP. =  136
ESP. PER BASE 2 =  8
VAL.BIN.PART.INT. SENZA SEGNO
11101010
VAL.BIN.PART.DEC. SENZA SEGNO
11000100100110111010011 0
NUMERO CALCOLATO =  234.768


NUMERO INTRODOTTO = -4321.987
QUATTRO BYTE MANTISSA IN MEMORIA
 135        15        229        96
QUATTRO BYTE MANTISSA IN BINARIO
10000111 00001111 11100101 01100000
STRINGA DI BIT INIZIALE
1000011100001111111001010110000 0
STRINGA BIT CON AGGIUNTO BIT INIZIALE
AL POSTO DEL BIT DI SEGNO
1000011100001111111001010110000 0
BYTE ESP. =  141
ESP. PER BASE 2 =  13
VAL.BIN.PART.INT. SENZA SEGNO
1000011100001
VAL.BIN.PART.DEC. SENZA SEGNO
1111110010101100000
NUMERO CALCOLATO = -4321.987
```

Il nome della VARIABILE SINGOLA REALE sta nei primi due byte; troviamo nel primo il codice ASCII della prima lettera e nel secondo il codice ASCII del secondo carattere o zero.

Il programma VARINMEM ci consente di vedere la rappresentazione in memoria di qualunque tipo di variabile. Esso definisce le seguenti variabili:
.singole: M, N, Y, Z, A, K, B%, I, C$, J, Y$,
.con indice: D, E%, F$.
Servendosi dei puntatori vengono stampati i contenuti delle diverse zone di memoria dedicate alle variabili.

```
1 REM VARINMEM
10 M=0:N=0:Y=0:Z=0
15 DIMD(6),E%(5,4),F$(5,3,2)
20 A=0:FORK=1TO50:A=A+1:NEXTK
25 FORK=0TO6:D(K)=K*3:NEXTK
30 B%=0:FORI=1TO50STEP2:B%=B%+2:NEXTI
35 FORK=0TO5:FORI=0TO4
40 E%(K,I)=K*1+9:NEXTI,K
45 C$="FINE"
50 FORK=0TO5:FORJ=0TO3:FORI=0TO2
55 READY$:F$(K,J,I)=C$+Y$
60 NEXTI,J,K
65 Y=PEEK(45)+256*PEEK(46)
70 OPEN4,4:CMD4
75 PRINT:PRINT"VARIABILI":M=Y
80 PRINTM;"**"
85 FORI=0TO10:FORN=0TO6
90 Y=PEEK(45)+256*PEEK(46)
95 PRINTPEEK(M+N);"  ";:NEXTN:PRINT:PRINT
100 M=M+7:NEXTI
105 Y=PEEK(47)+256*PEEK(48)
110 Z=PEEK(49)+256*PEEK(50)
115 PRINT:PRINT"ARRAY":PRINTY;"**":K=Y
120 M=PEEK(K+2)+256*PEEK(K+3)
125 I=0:FORJ=KTOK+M-1
130 PRINTPEEK(J);" ";:I=I+1
```

```
135 IFI=7THENPRINT:I=0
140 NEXTJ:K=K+M:IFK=ZTHEN150
145 PRINT:GOTO120
150 PRINT:PRINT"FINE ARRAY: ";PEEK(Z)
155 PRINT:PRINT:PRINT"CORPO STRINGHE":PRINT
160 Y=PEEK(51)+256*PEEK(52)
165 Z=PEEK(55)+256*PEEK(56)
170 I=0:FORK=YTOY+50:PRINTK;"*";PEEK(K);
175 I=I+1:IFI=3THENPRINT:I=0
180 NEXTK
185 I=0:FORK=Z-50TOZ:PRINTK;"*";PEEK(K);
190 I=I+1:IFI=3THENPRINT:I=0
195 NEXTK
200 PRINT#4:CLOSE4:STOP
205 DATAAAA,BBB,CCC,DDD,EEE,FFF,GGG,III,LLL
210 DATAA,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T
215 DATAU,V,W,X,Y,Z,A1,B1,C1,D1,E1,F1,G1,H1,I1
220 DATAJ1,K1,L1,M1,N1,O1,P1,Q1,R1,S1,T1,U1,V1
225 DATAW1,X1,Y1,Z1,ABC,DEF,GHI,JKL,MNO,PQR,STU
230 DATAVWX,YZZ,PIPPO,PLUTO
```

Nella memoria destinata alle variabili del programma BASIC esistono tre zone distinte; esse sono:

.variabili singole, subito dopo il programma, puntatore in 45 e 46,

.variabili con indice, dopo le variabili singole, puntatore in 47 e 48,

.corpi delle stringhe, a partire dal fondo della memoria, per indirizzi decrescenti, puntatori in 51/ 52 e in 53/54.

Le stringhe sono rappresentate in due parti, la testata, con il nome, la lunghezza in caratteri, il puntatore al corpo della stringa e le altre indicazioni necessarie per le variabili stringa con indice, sta o nella prima o nella seconda zona, mentre il corpo della stringa, cioe' i caratteri, stanno nella terza zona. Quando il contenuto di una variabile stringa varia, si ha l'aggiornamento della testata e del corpo. Dal momento che esiste questa separazione in zone

delle variabili, quando viene creata una variabile singola, le variabili con indice gia' esistenti vengono traslate in memoria per far posto alla nuova variabile.
Nel programma VARINMEM vengono create le variabili, viene letto con la funzione PEEK il contenuto e stampato.

RISULTATI PROGRAMMA VARINMEM

```
VARIABILI
 5159 **
 77    0    141    33    56    0    0

 78    0    130    64    0    0    0

 89    0    141    33    56    0    0

 90    0    0    0    0    0    0

 65    0    134    72    0    0    0

 75    0    131    64    0    0    0

 194    128    0    50    0    0    0

 73    0    131    96    0    0    0

 67    128    4    240    63    0    0

 74    0    131    0    0    0    0

 89    128    5    159    60    0    0
```

```
ARRAY
 5236 **
 68    0    42    0    1    0    7
 0    64    0    0    0    130    64
 0    0    0    131    64    0    0
 0    132    16    0    0    0    132
 64    0    0    0    132    112    0
 0    0    133    16    0    0    0

 197    128    69    0    2    0    5
 0    6    0    9    0    10    0
 11    0    12    0    13    0    14
 0    9    0    10    0    11    0
 12    0    13    0    14    0    9
 0    10    0    11    0    12    0
 13    0    14    0    9    0    10
 0    11    0    12    0    13    0
 14    0    9    0    10    0    11
 0    12    0    13    0    14
 70    128    227    0    3    0    3
 0    4    0    6    7    226    63
 5    74    63    5    210    62    6
 86    62    6    198    61    6    54
 61    7    184    63    5    44    63
 5    180    62    6    50    62    6
 162    61    7    12    61    7    142
 63    5    14    63    5    150    62
 6    14    62    6    126    61    7
 226    60    5    104    63    5    240
 62    5    120    62    6    234    61
 6    90    61    7    184    60    7
 212    63    5    64    63    5    200
 62    6    74    62    6    186    61
 7    40    61    7    170    63    5
 34    63    5    170    62    6    38
 62    6    150    61    7    254    60
 7    128    63    5    4    63    5
 140    62    6    2    62    6    114
 61    7    212    60    5    94    63
 5    230    62    5    110    62    6
 222    61    6    78    61    9    166
 60    7    198    63    5    54    63
```

```
 5   190   62   6   62   62   6
 174   61   7   26   61   7   156
 63   5   24   63   5   160   62
 6   26   62   6   138   61   7
 240   60   7   114   63   5   250
 62   5   130   62   6   246   61
 6   102   61   7   198   60   5
 84   63   5   220   62   6   98
 62   6   210   61   6   66   61
 9   148   60
FINE ARRAY:  255
```

CORPO STRINGHE

```
 15508 * 70  15509 * 73  15510 * 78
 15511 * 69  15512 * 80  15513 * 76
 15514 * 85  15515 * 84  15516 * 79
 15517 * 195  15518 * 21  15519 * 80
 15520 * 76  15521 * 85  15522 * 84
 15523 * 79  15524 * 111  15525 * 20
 15526 * 70  15527 * 73  15528 * 78
 15529 * 69  15530 * 80  15531 * 73
 15532 * 80  15533 * 80  15534 * 79
 15535 * 123  15536 * 21  15537 * 80
 15538 * 73  15539 * 80  15540 * 80
 15541 * 79  15542 * 5  15543 * 255
 15544 * 70  15545 * 73  15546 * 78
 15547 * 69  15548 * 89  15549 * 90
 15550 * 90  15551 * 51  15552 * 21
 15553 * 89  15554 * 90  15555 * 90
 15556 * 3  15557 * 255  15558 * 70
 16324 * 3  16325 * 255  16326 * 70
 16327 * 73  16328 * 78  16329 * 69
 16330 * 67  16331 * 67  16332 * 67
 16333 * 126  16334 * 21  16335 * 67
 16336 * 67  16337 * 67  16338 * 3
 16339 * 255  16340 * 70  16341 * 73
 16342 * 78  16343 * 69  16344 * 66
 16345 * 66  16346 * 66  16347 * 54
 16348 * 21  16349 * 66  16350 * 66
 16351 * 66  16352 * 3  16353 * 255
```

```
16354 * 70  16355 * 73  16356 * 78
16357 * 69  16358 * 65  16359 * 65
16360 * 65  16361 * 238  16362 * 20
16363 * 65  16364 * 65  16365 * 65
16366 * 3  16367 * 255  16368 * 70
16369 * 73  16370 * 78  16371 * 69
16372 * 97  16373 * 20  16374 * 255
```

La prima parte dei risultati ci mostra, dopo la parola VARIABILI, il valore del puntatore alle variabili, che qui e' il numero 5159. Questo significa che il programma VARINMEM occupa la parte di memoria che va da 4097 a 5158. A partire da 5159 troviamo le 11 variabili singole del programma; ognuna occupa 7 byte. Troviamo ordinatamente:

M, nome 77 0, caratteristica 141, mantissa 33 56 0 0.

N, nome 78 0, caratteristica 130, mantissa 64 0 0 0.

Y, nome 89 0, caratteristica 141, mantissa 33 56 0 0.

Z, nome 90 0, caratteristica e mantissa 0, dato che al momento della sua lettura contiene 0;

A, nome 65 0, caratteristica 134, mantissa 72 0 0 0.

K, nome 75 0, caratteristica 131, mantissa 64 0 0 0.

B%, nome 194 128, valore intero 0 50;

I, nome 73 0, caratteristica 131, mantissa 96 0 0 0.

C$, nome 67 128, cioe' il codice ASCII del primo carattere del nome e il codice ASCII del secondo carattere del nome + 128 a significare il suffisso $; il terzo byte contiene il contatore dei caratteri, che puo' essere al massimo 255 e qui e' 4, infatti C$ contiene FINE, che le viene assegnato alla linea 45 del programma. Il quarto e quinto byte contengono il puntatore al corpo della stringa nel formato byte basso e byte

alto; qui abbiamo 240 e 63, che danno il numero 16368=63*256+240. Nella zona corpo delle stringhe troviamo da 16368 a 16371 la parola FINE.
J, nome 74 0, caratteristica 131, mantissa 0 0 0 0.
Y$, nome 89 128, numero caratteri 5, puntatore a 60*256+159=15519. Da 15519 a 15523 si trova la parola PLUTO, ultima letta in Y$.
Andando a cercare nella terza zona dei risultati, quella denominata CORPO STRINGHE, troviamo da 16368 a 16371 la parola FINE seguita da due byte che contengono il puntatore (20*256+97=5217) al byte che contiene il numero di caratteri della stringa nella sua testata. Analogamente per la stringa Y$, troviamo da 15519 a 15523 la parola PLUTO, seguita dal puntatore (20*256+111=5231) al byte che da' la lunghezza della stringa nella sua testata.

Passando alla seconda zona dei risultati troviamo le variabili con indice. Dopo la parola ARRAY (denominazione inglese delle variabili con indice, dette anche MATRICI), seguita dall'indirizzo di inizio delle stesse nel byte 5236.
Seguono ordinatamente:
D(6), cioe' la variabile numerica D di 7 elementi reali. Nei primi 2 byte compare il nome, 68 0; seguono due byte, byte basso e byte alto, che danno il numero totale dei byte occupati, in questo caso 42. Segue un byte con il numero delle dimensioni, 1 in questo caso. Si hanno poi due byte, byte alto e byte basso, contenenti il numero di elementi dell'unica dimensione, 7 in questo caso. Seguono tanti gruppi di 5 byte quanti sono gli elementi, contenenti i numeri reali nel formato caratteristica e mantissa.
E%(5,4), cioe' la variabile numerica intera con indice E% di 30 elementi (6*5). Vediamo 2 byte per il nome, 197 128; seguono i due byte

contatori della memoria occupata, 69 byte qui. Segue il byte con il numero delle dimensioni, 2, il numero di elementi dell'ultima dimensione, 0 5, e infine il numero di elementi della prima dimensione, 0 6. Gli elementi che seguono occupano ciascuno 2 byte, dato che si tratta di numeri interi. Come vedi nel caso delle variabili con indice intere si ha un risparmio di memoria rispetto alle singole.
F$(5,3,2), cioe' la variabile stringa con indici F$, di 72 elementi. Per la testata valgono le stesse regole valide per le altre variabili con indice. Per gli elementi sono usati 3 byte; il primo dice quanto e' lungo l'elemento e gli ultimi due sono il puntatore al corpo della stringa.
Nella zona ARRAY, la descrizione di D(6) inizia alla prima riga e occupa 6 righe. La descrizione di E%(5,4) inizia alla settima riga e termina alla sedicesima riga. La descrizione di F$(5,3,) inizia alla diciassettesima riga (70 128 227 0 3 0 3); se conti 11 numeri, cioe' passi al quinto numero della riga seguente trovi: 7 226 63, cioe' il primo elemento F$(0,0,0), che contiene la parola FINEAAA; esso si trova da 16354 a 16360 e in 16361 e 16362 si trova il puntatore alla sua lunghezza nella testata; troviamo il numero 5358=20*256+238, cioe' l'indirizzo del byte che contiene 7.

La zona CORPO STRINGHE non e' stata stampata tutta, ma solo una parte all'inizio, da 16374 in su, e una parte alla fine, da 15508 in giu'.
Se ti interessa, con un po' di pazienza puoi ritrovare i dati che desideri. L'unico problema e' calcolare bene i valori dati da due byte, ricordando quando il byte alto precede quello basso e quando no.

Non ci occupiamo qui dell'occupazione di memoria da parte del SISTEMA OPERATIVO e dell'INTERPRETE BASIC e dei programmi in BASIC, argomenti per i quali rimandiamo al secondo volume.

APPENDICE D

VALORE DELLE NOTE

Questa appendice contiene la lista delle 72 note che formano 6 ottave, il valore che bisogna usare come secondo parametro nell'istruzione SOUND per ottenere la nota, e la frequenza della nota espressa in Hertz.

NOTA	VALORE	FREQUENZA
LA	7	109.97
LA#	64	116.50
SI	118	123.44
DO	169	130.81
DO#	217	138.59
RE	262	146.77
RE#	305	155.55
MI	345	164.71
FA	383	174.48
FA#	419	184.86
SOL	453	195.87
SOL#	485	207.50
LA	516	220.16
LA#	544	233.00
SI	571	246.89
DO	597	261.92
DO#	621	277.52
RE	643	293.54
RE#	665	311.53
MI	685	329.91
FA	704	349.50
FA#	722	370.33
SOL	739	392.42
SOL#	755	415.76

NOTA	VALORE	FREQUENZA
LA	770	440.32
LA#	784	466.00
SI	798	494.87
DO	810	522.62
DO#	822	553.67
RE	834	588.63
RE#	844	621.34
MI	854	657.89
FA	864	699.00
FA#	873	740.67
SOL	881	782.10
SOL#	889	828.45
LA	897	880.63
LA#	904	932.00
SI	911	989.74
DO	917	1045.24
DO#	923	1107.33
RE	929	1177.27
RE#	934	1242.67
MI	939	1315.77
FA	944	1398.01
FA#	948	1471.58
SOL	953	1575.22
SOL#	957	1669.26

NOTA	VALORE	FREQUENZA
LA	960	1747.51
LA#	964	1864.01
SI	967	1962.11
DO	971	2110.20
DO#	974	2236.81
RE	976	2330.01
RE#	979	2485.34
MI	982	2662.87
FA	984	2796.01
FA#	986	2943.17
SOL	988	3106.68
SOL#	990	3289.43
LA	992	3495.01
LA#	994	3728.02
SI	996	3994.30
DO	997	4142.24
DO#	999	4473.62
RE	1000	4660.02
RE#	1002	5083.66
MI	1003	5325.74
FA	1004	5592.02
FA#	1005	5886.34
SOL	1006	6213.36
SOL#	1007	6578.85

APPENDICE E

MESSAGGI DI ERRORE

==========
errore 1: TOO MANY FILES

Questo messaggio appare quando si cerca di aprire l'undicesimo file. Il COMMODORE 16 infatti puo' tenere aperti al massimo dieci file contemporaneamente.

==========
errore 2: FILE OPEN

Questo messaggio appare quando si vuole aprire un file logico che gia' e' stato aperto.

==========
errore 3: FILE NOT OPEN

Si cerca di comunicare con un file prima di averlo aperto.

==========
errore 4: FILE NOT FOUND

Il file richiesto non e' presente sul dischetto, o lo sportello del drive non e' stato chiuso, oppure il drive e' guasto. Nelle operazioni su cassetta questo messaggio appare dopo che e' stato incontrato sul nastro il segnale "fine nastro"

errore 5: DEVICE NOT PRESENT

La periferica con cui si e' cercato di comunicare non e' collegata, oppure e' spenta, oppure e' guasta.

errore 6: NOT INPUT FILE

Si cerca di ricevere dati da un file che non e' stato aperto in lettura (ad esempio un file su nastro aperto in scrittura).

errore 7: NOT OUTPUT FILE

Si e' cercato di inviare dati a un file aperto in lettura (ad esempio la tastiera).

errore 8: MISSING FILE NAME

Si e' cercato di salvare un programma su disco senza specificarne il nome.

errore 9: ILLEGAL DEVICE NUMBER

Si e' cercato di eseguire operazioni di ingresso-uscita non consentite per la periferica individuata; ad esempio salvare su unita' 3, che e' il video.

errore 10: NEXT WITHOUT FOR

E' stato incontrato NEXT prima di FOR. Fai molta attenzione a chiudere bene tutti i cicli FOR..NEXT: vedi al proposito il Capitolo 6

==========
errore 11: SYNTAX ERROR

E' stato introdotto un comando che il calcolatore non riconosce: controlla bene tutte le lettere della frase introdotta: l'errore di sintassi e' dovuto spesso a errori di battuta. Questo errore appare anche quando si cerca di assegnare dei valori alle variabili riservate, come TI, TI$, ST, DS, DS$, ER, EL. Anche l'uso di parole chiave (GO, TO, IF, OR, ON, ecc) come variabili produce SYNTAX ERROR. Esempio: AGO=21 e' sbagliato perche' AGO viene interpretato come A + GO, e GO e' una parola riservata. A volte puo' capitare questo errore in una linea che appare perfetta: puo' capitare che l'interprete abbia interperetato male delle parole-chiave: ad esempio, la linea:

IFSTAND64THENPRINT"FINE FILE":EXIT

produce SYNTAX ERROR, perche' viene interpretata cosi'.

IF S TAN D64 THEN PRINT.....

(TAN e' una parola riservata) Una linea cosi' e' palesemente sbagliata. Perche' questa linea sia interpretata correttamente occorre porre gli spazi nel modo seguente:

IF ST AND 64 THEN ....

Se pensi che il tuo SYNTAX ERROR sia causato da una situazione di questo tipo, puoi per sicurezza separare tutte le parole chiave con spazi.

==========
errore 12: RETURN WITHOUT GOSUB

E' stata incontrata un'istruzione RETURN senza aver incontrato GOSUB: questo errore capita spesso quando ci si dimentica di porre il comando END alla fine del programma principale, e il calcolatore prosegue, andando ad eseguire le subroutine che seguono il programma.

==========
errore 13: OUT OF DATA

Questo errore l'hai trovato sicuramente tutte le volte che hai premuto RETURN quando il cursore si trovava su una linea che iniziava con la scritta "READY.". Il COMMODORE 16 interpreta questo messaggio come il comando READ Y e cerca nel programma una linea DATA. Se non la trova segnala questo errore. Nel programma invece questo errore significa che le linee DATA sono state lette tutte, e si e' tentato di leggere piu' dati di quanti sono disponibili.

==========
errore 14: ILLEGAL QUANTITY

Una variabile ha superato il limite consentito per l'operazione richiesta. Aiutati con la funzione HELP e chiedi il valore delle variabili che interessano la linea dove e' avvenuto l'errore; eventualmente controlla i limiti consentiti alle variabili nel comando che lampeggia dopo HELP.

==========
errore 15: OVERFLOW

Una variabile ha raggiunto un valore troppo alto o troppo basso. Leggi il Capitolo 3 per sapere quali sono i valori massimi e minimi delle variabili.

==========
errore 16: OUT OF MEMORY

Non c'e' posto in memoria per eseguire l'operazione richiesta: per rimediare puoi acquistare un'espansione di memoria, o migliorare la compattazione del tuo programma ponendo molti comandi su una sola linea, o, a volte, eseguire l'istruzione GRAPHIC CLR, e non usare la pagina

grafica, oppure ancora dividere il programma in pezzettini (se hai il drive) e caricare di volta in volta le routine che ti servono. Questo messaggio puo' apparire anche quando lo STACK (particolare area di lavoro usata dalle routine del sistema) viene riempito, cioe' quando ci sono troppi FOR..NEXT uno dentro l'altro, con dei GOSUB, con dei DO...LOOP, o con delle espressioni matematiche con molti livelli di parentesi. Prova ad eseguire il programma:

```
10 GOSUB 10
```

esso uscira' con il messaggio OUT OF MEMORY: al BASIC infatti sono riservati 128 bytes di memoria per lo STACK, che non possono essere espansi. Per capire se questo messaggio e' stato causato da un riempimento dello STACK, puoi chiedere quanta memoria e' ancora libera, mediante il comando:

```
PRINT FRE(0)
```

Se il numero e' grande, allora e' probabile che la memoria piena sia lo STACK, se e' piccolo allora la memoria piena dovrebbe essere quella utente. Per avere la certezza sulla causa del messaggio puoi usare la funzione HELP, e vedere la natura del comando che ha causato l'errore. Se il comando era un DIM, o un assegnamento di variabile, allora la memoria piena e' quella utente, se invece il comando e' un GOSUB, o un FOR, o un DO, o il calcolo di un'espressione con delle parentesi, allora la memoria piena e' quella riservata allo STACK. In quest'ultimo caso devi provvedere a cambiare la struttura del tuo programma, in modo da occupare meno posizioni nello STACK.

==========

errore 17: UNDEF'D STATEMENT

Hai eseguito GOTO ,GOSUB, RUN, RESTORE, TRAP o RESUME con il numero di una linea che non esiste.

==========
errore 18: BAD SUBSCRIPT

L'indice che hai usato per un variabile con indice supera il valore indicato in sede di dimensionamento, oppure e' negativo.

==========
errore 19: REDIM'D ARRAY

Il calcolatore ha incontrato due volte l'istruzione DIM sulla stessa variabile; questo puo' essere dovuto a un effettivo doppio dimensionamento, oppure al fatto che il BASIC dimensiona automaticamente una variabile con indice a 10, quando si fa riferimento a un suo elemento la prima volta. Ad esempio il comando:
PRINT A(4)
fa si' che venga eseguita una DIM A(10) automaticamente, se il vettore A non era ancora stato dimensionato, e cio' produce questo errore se successivamente si incontra un'istruzione DIM. Se si vuole effettivamente cancellare un vettore, e dimensionarlo nuovamente, allora bisogna eseguire prima una CLR, che cancella tutte le variabili, e poi dimensionare nuovamente il vettore. Ma attenzione: l'istruzione CLR cancella tutto lo STACK, quindi non puo' essere eseguita ne' in una subroutine, ne' in un ciclo.

==========
errore 20: DIVISION BY ZERO

Il divisore di una divisione e' stato di valore 0. Per evitare l'arresto del programma devi controllare il divisore, ed evitare la linea con il calcolo se questo vale zero. Utile in certi casi puo' essere l'istruzione TRAP, che ti permette di eseguire una subroutine speciale dopo un errore; cosi' facendo puoi evitare di calcolare a parte il denominatore, rendendo piu' veloce l'esecuzione del programma.

==========
errore 21: ILLEGAL DIRECT

Alcuni comandi non possono essere eseguiti in modo diretto, ma solo in un programma, come INPUT e GET.

==========
errore 22: TYPE MISMATCH

Il tipo di variabile usato non concorda con quello richiesto; assegnare ad una variabile numerica un valore alfabetico produce questo errore.

==========
errore 23: STRING TOO LONG

Le stringhe possono avere la lunghezza massima di 255 caratteri. Questo messaggio appare facilmente col comando SSHAPE, e appare anche quando introduci linee di comando che superano gli 88 caratteri. Tieni presente che conviene non superare gli 80 caratteri in una linea BASIC, in modo che i listati sulla stampante non debbano andare a capo, e le eventuali correzioni siano piu' facili, avendo a disposizione alcuni caratteri.

==========
errore 24: FILE DATA

I dati letti dal file non sono accettati.

==========
errore 25: FORMULA TOO COMPLEX

Questo messaggio appare di solito quando il calcolatore e' in "tilt", a causa di POKE o SYS o routine in linguaggio macchina sbagliate.

==========
errore 26: CAN'T CONTINUE

Il programma puo' riprendere dopo uno STOP, un END, o un BREAK ottenuto premendo il tasto RUN/STOP, a patto di non aver ne' introdotto ne' modificato linee BASIC. Se il programma si ferma per un errore, allora il comando CONT provoca sempre questo messaggio di errore.

==========
errore 27: UNDEF'D FUNCTION

Si e' cercato di calcolare il valore di una funzione non definita.

==========
errore 28: VERIFY ERROR

Il programma che si trova in memoria e' diverso da quello che si trova su nastro o disco: questo errore puo' significare che un file e' stato salvato su un supporto labile, o piu' facilmente che il programma in memoria sia stato leggermente cambiato.

==========
errore 29: LOAD ERROR

Un programma che si trova su nastro non si riesce a caricare: prova a pulire le testine del registratore; se l'errore rimane, e sei sicuro che la cassetta e' stata registrata da un registratore in ordine, allora puoi portare il tuo registratore a un servizio assistenza tecnica, e richiedere di allineare le testine.

==========
errore 30: BREAK ERROR

Hai premuto il tasto RUN/STOP durante l'esecuzione di un programma, oppure hai posto un

comando STOP nel programma; se non vuoi che questo messaggio appaia, usa il comando END al posto di STOP.

==========

errore 31: CAN'T RESUME

Il comando RESUME puo' essere eseguito solamente in una routine in cui si e' entrati a causa di un errore, con l'istruzione TRAP. Questo errore appare quando RESUME e' stato incontrato senza l'errore.

==========

errore 32: LOOP NOT FOUND

I cicli DO...LOOP devono terminare con un LOOP, altrimenti viene segnalato questo errore al momento di uscire dal loop.

==========

errore 33: LOOP WITHOUT DO

Questo errore e' molto simile a NEXT WITHOUT FOR; l'istruzione LOOP non ha senso se prima non si e' incontrato un DO.

==========

errore 34: DIRECT MODE ONLY

Alcuni comandi si possono eseguire solo in modo diretto, cioe' non da programma; AUTO, RENUMBER, DELETE sono tra questi.

==========

errore 35: NO GRAPHICS AREA

Si e' cercato di eseguire comandi grafici prima che sia stata riservata memoria all'area grafica con il comando GRAPHIC.

==========

errore 36: BAD DISK

Questo errore appare quando si cerca di eseguire operazioni sul disco, e queste non hanno buon esito; richiedi il valore di DS$ per sapere la natura dell'errore.

APPENDICE F

FUNZIONI MATEMATICHE DERIVATE

Questa appendice mostra come calcolare le funzioni che non sono direttamente richiamabili dal BASIC 3.5.

FUNZIONE	EQUIVALENTE BASIC
sec(X)	1/COS(X)
cosec(X)	1/SIN(X)
cotan(X)	1/TAN(X)
arcsin(X)	ATN(X/SQR(-X*X+1))
arccos(X)	ATN(X/SQR(-X*X+1))+π/2
arcsec(X)	ATN(X/SQR(X*X-1))
arccosec(X)	ATN(X/SQR(X*X-1))+ (SGN(X)-1)*π/2
arccotg(X)	ATN(X)+π/2
sinh(X)	(EXP(X)-EXP(-X))/2
cosh(X)	(EXP(X)+EXP(-X))/2
tgh(X)	EXP(-X)/(EXP(X)+ EXP(-X))*2+1
sech(X)	2/(EXP(X)+EXP(-X))
cosech(X)	2/(EXP(X)-EXP(-X))
cotgh(X)	EXP(-X)/(EXP(X)- EXP(-X))*2+1

FUNZIONE	EQUIVALENTE BASIC
arcsinh(X)	LOG(X+SQR(X*X+1))
arccosh(X)	LOG(X+SQR(X*X-1))
arctgh(X)	LOG((1+X)/(1-X))/2
arcsech(X)	LOG(SQR(-X*X+1)+1/X)
arccosch(X)	LOG(SGN(X)*SQR(X*X+ 1/X))
arccotgh(X)	LOG((X+1)/(X-1))/2

APPENDICE G

CONVERSIONI DEC./ESADEC./BIN.

Nella tabella che segue sono riportati i numeri decimali da 0 a 255 e i loro corrispondenti esadecimali.

DEC.	HEX.	DEC.	HEX.	DEC.	HEX.	DEC.	HEX.
0	00	32	20	64	40	96	60
1	01	33	21	65	41	97	61
2	02	34	22	66	42	98	62
3	03	35	23	67	43	99	63
4	04	36	24	68	44	100	64
5	05	37	25	69	45	101	65
6	06	38	26	70	46	102	66
7	07	39	27	71	47	103	67
8	08	40	28	72	48	104	68
9	09	41	29	73	49	105	69
10	0A	42	2A	74	4A	106	6A
11	0B	43	2B	75	4B	107	6B
12	0C	44	2C	76	4C	108	6C
13	0D	45	2D	77	4D	109	6D
14	0E	46	2E	78	4E	110	6E
15	0F	47	2F	79	4F	111	6F
16	10	48	30	80	50	112	70
17	11	49	31	81	51	113	71
18	12	50	32	82	52	114	72
19	13	51	33	83	53	115	73
20	14	52	34	84	54	116	74
21	15	53	35	85	55	117	75
22	16	54	36	86	56	118	76
23	17	55	37	87	57	119	77
24	18	56	38	88	58	120	78
25	19	57	39	89	59	121	79

TABELLA G.1 Conversione decimale/esadecimale

26	1A	58	3A	90	5A	122	7A
27	1B	59	3B	91	5B	123	7B
28	1C	60	3C	92	5C	124	7C
29	1D	61	3D	93	5D	125	7D
30	1E	62	3E	94	5E	126	7E
31	1F	63	3F	95	5F	127	7F
DEC.	HEX.	DEC.	HEX.	DEC.	HEX.	DEC.	HEX.
128	80	160	A0	192	C0	224	E0
129	81	161	A1	193	C1	225	E1
130	82	162	A2	194	C2	226	E2
131	83	163	A3	195	C3	227	E3
132	84	164	A4	196	C4	228	E4
133	85	165	A5	197	C5	229	E5
134	86	166	A6	198	C6	230	E6
135	87	167	A7	199	C7	231	E7
136	88	168	A8	200	C8	232	E8
137	89	169	A9	201	C9	233	E9
138	8A	170	AA	202	CA	234	EA
139	8B	171	AB	203	CB	235	EB
140	8C	172	AC	204	CC	236	EC
141	8D	173	AD	205	CD	237	ED
142	8E	174	AE	206	CE	238	EE
143	8F	175	AF	207	CF	239	EF
144	90	176	B0	208	D0	240	F0
145	91	177	B1	209	D1	241	F1
146	92	178	B2	210	D2	242	F2
147	93	179	B3	211	D3	243	F3
148	94	180	B4	212	D4	244	F4
149	95	181	B5	213	D5	245	F5
150	96	182	B6	214	D6	246	F6
151	97	183	B7	215	D7	247	F7
152	98	184	B8	216	D8	248	F8
153	99	185	B9	217	D9	249	F9
154	9A	186	BA	218	DA	250	FA
155	9B	187	BB	219	DB	251	FB
156	9C	188	BC	220	DC	252	FC
157	9D	189	BD	221	DD	253	FD
158	9E	190	BE	222	DE	254	FE
159	9F	191	BF	223	DF	255	FF

TABELLA G.1 Conversione decimale/esadecimale (continuazione)

Segue il programma CONV1 che stampa una tabellina contenente i numeri decimali da 0 a 15 e i corrispondenti numeri esadecimali e binari.

```
1 REM CONV1
6 OPEN4,4:CMD4
11 DATA0,0000,1,0001,2,0010,3,0011,4,0100
16 DATA5,0101,6,0110,7,0111,8,1000,9,1001
21 DATAA,1010,B,1011,C,1100,D,1101,E,1110,F,1111
26 PRINT"DECIMALE  ESADECIMALE  BINARIO"
31 PRINT
36 FORK=0TO15
41 READA$,B$:C$=STR$(K)
46 C$=RIGHT$("  "+C$,2)
51 PRINT"   ";C$;"          ";A$;"        ";B$
56 NEXTK
61 PRINT#4:CLOSE4
```

RISULTATI PROGRAMMA CONV1

DECIMALE	ESADECIMALE	BINARIO
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Nel programma CONV1 i numeri corrispondenti ai decimali, in esadecimale e binario, sono ottenuti dal file interno di dati generato dalle frasi DATA delle linee 11, 16 e 21, nelle quali sono riportati a coppie il numero in esadecimale e il corrispondente in binario.

Il COMMODORE 16 dispone di due funzioni in BASIC, la DEC e la HEX$, che consentono di operare conversioni da esadecimale in decimale e viceversa per numeri esadecimali di al massimo 4 cifre e per numeri decimali compresi tra 0 e 65535.
Il programma CONV2 che segue accetta come dato di ingresso una stringa esadecimale di al massimo 4 caratteri, e la converte nel numero decimale corrispondente.

```
1 REM CONV2
10 PRINT"[10]SCRIVI UN NUMERO ESADECIMALE DI"
15 PRINT"AL MASSIMO 4 CIFRE"
20 INPUTN$
25 IFLEN(N$)>4THEN10
30 FORK=1TOLEN(N$)
35 A$=MID$(N$,K,1)
40 IFA$>="0"ANDA$<="9"ORA$>="A"ANDA$<="F"THEN50
45 GOTO10
50 NEXTK
60 PRINTN$;" IN DECIMALE = ";DEC(N$)
65 STOP
```

Nel programma CONV2 la stringa esadecimale da convertire viene controllata e accettata solo se e' di al massimo 4 caratteri e i caratteri sono cifre esadecimali, cioe' comprese negli intervalli 0-9 e A-F.

Il programma CONV3 che segue accetta in ingresso un numero decimale compreso tra 0 e 65535 e lo

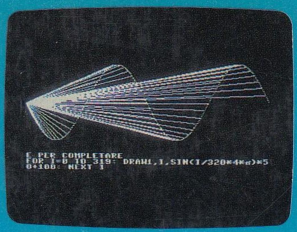
converte nella stringa esadecimale corrispondente.

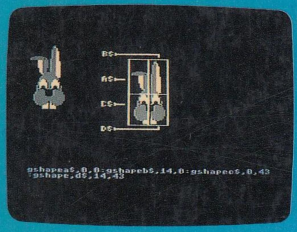
```
1 REM CONV3
5 PRINT"[U]SCRIVI UN NUMERO DECIMALE POSITIVO"
10 PRINT"COMPRESO TRA 0 E 65535"
15 INPUTN
20 IFN<0ORN>65535THEN5
25 PRINTN;" IN ESADECIMALE = ";HEX$(N)
30 STOP
```

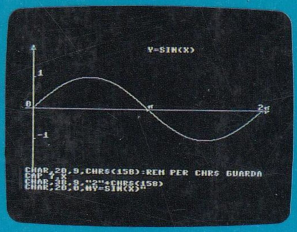
Il programma CONV3 accetta in ingresso un numero positivo compreso tra 0 e 65535 e lo converte sempre in una stringa di 4 caratteri esadecimali.

Il programma CONV4 che segue consente di convertire un numero decimale, compreso tra 0 e 255, in binario; dopo averlo convertito e stampato, viene effettuata la controprova, cioe' il numero binario viene riconvertito in decimale e stampato. In questo caso le conversioni vengono operate effettuando dei calcoli.

```
1 REM CONV4
5 PRINT"[U]SCRIVI UN NUMERO DECIMALE POSITIVO"
10 PRINT"COMPRESO TRA 0 E 255"
15 INPUTN
20 IFN<0ORN>255THEN5
25 X$="":A=N
30 FORK=7TO0STEP-1
35 IFN>=2↑KTHENX$=X$+"1":N=N-2↑K:GOTO45
40 X$=X$+"0"
45 NEXTK
50 PRINTA;" IN BINARIO = ";X$:PRINT
55 A=0:FORK=8TO1STEP-1
60 A$=MID$(X$,K,1)
65 IFA$="1"THENA=A+2↑(8-K)
70 NEXTK
75 PRINTX$;" IN DECIMALE = ";A
80 STOP
```







Il libro si propone di insegnare ad usare il COMMODORE 16 e a programmarlo in BASIC servendosi della tastiera, del video e del registratore a cassetta.
La cassetta contiene la presentazione degli argomenti che sono trattati nei capitoli da 1 a 10; i contenuti del libro sono parzialmente anticipati nella presentazione su video, e la lettura risulta quindi facilitata.
Il Capitolo 0 presenta il libro e insegna ad usare la cassetta. Seguono 10 capitoli, ognuno dei quali va letto dopo aver seguito il relativo programma di presentazione sul video.
Il Capitolo 1 riguarda l'uso della tastiera. Il Capitolo 2 mostra le caratteristiche del video. Il Capitolo 3 insegna a usare il COMMODORE 16 come calcolatrice. Con i Capitoli 4 e 5 si incomincia a imparare a programmare usando le istruzioni più semplici del BASIC. Nel Capitolo 6 si trattano le strutture condizionali e i cicli. Il Capitolo 7 riguarda la grafica ad alta risoluzione e multicolore. Il Capitolo 8 tratta l'animazione e il Capitolo 9 il suono. Nel Capitolo10 si descrive l'uso di un buon numero di istruzioni del BASIC. Il Capitolo 11 riguarda operazioni avanzate e termina con la presentazione di un programma per la gestione di un archivio di dati su cassetta.
Numerose appendici permettono di approfondire la conoscenza del calcolatore; in particolare, l'Appendice A contiene la scheda completa del BASIC 3.5, e l'Appendice E tratta in modo approfondito le segnalazioni di errore da parte del sistema.

L. 35.000

COD 413B ISBN88-7056-196-8